This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

Hierarchical Contextual Attention Recurrent Neural Network for Map Query Suggestion

Jun Song, Jun Xiao, Fei Wu, Haishan Wu, Tong Zhang, Zhongfei Zhang, Wenwu Zhu, *Fellow, IEEE*

Abstract—The query logs from an on-line map query system provide rich cues to understand the behaviours of human crowds. With the growing ability of collecting large scale query logs, the query suggestion has been a topic of recent interest. In general, query suggestion aims at recommending a list of relevant queries w.r.t. users' inputs via an appropriate learning of crowds' query logs. In this paper, we are particularly interested in map query suggestions (e.g., the predictions of location-related queries) and propose a novel model *Hierarchical Contextual Attention Recurrent Neural Network* (HCAR-NN) for map query suggestion in an encoding-decoding manner. Given crowds map query logs, our proposed HCAR-NN not only learns the local temporal correlation among map queries in a query session (e.g., queries in a short-term interval are relevant to accomplish a search mission), but also captures the global longer range contextual dependencies among map query sessions in query logs (e.g., how a sequence of queries within a short-term interval has an influence on another sequence of queries). We evaluate our approach over millions of queries from a commercial search engine (i.e., *Baidu Map*). Experimental results show that the proposed approach provides significant performance improvements over the competitive existing methods in terms of classical metrics (i.e. *Recall@K* and *MRR*) as well as the prediction of crowds' search missions.

Index Terms—Recurrent Neural Network, Long-Short Term Memory, Map Query Prediction, Search Mission

1 Introduction

NOWADAYS, billions of people readily use on-line map searching services like *Google Map* or *Baidu Map* in daily life. With the rapid growth of users and the enhancing ability of collecting query logs, map query suggestion has been a topic of recent interest to understand the crowds' location-relevant behaviours.

Generally speaking, a user's query record can be partitioned into couples of query sessions, i.e., sequences of submitted queries by an unique user within a short time interval. The queries in a given query session have strong temporal correlations since they are reformulated to complete the same search mission (i.e., searching for a good restaurant for a dinner party). Moreover, since crowd routine behaviours are geographically related, a previous map query session by a user often has a contextual dependency on a subsequent map query session of the same user (e.g., a map query about a hotel probably leads to a subsequent map query about a nearby restaurant). In real life, a user's query action may always be interrupted by an un-relevant event (e.g., a tourist may want to find a toilet while searching the place he plans to visit). Therefore, appropriately capturing the contextual correlations between neighbouring query sessions is crucial to secure a good performance of query suggestion.

In recent years, in both academia and industry, similar problems (e.g., query reformulation and query suggestion) are drawn with more and more attention. Factor models [1], [2], [3], [4], which treat the suggestion problem as a matrix reconstruction problem, are the most common methods used in suggestion systems. The rating or counting based factor models decompose the sparse user-item rating/counting matrix into a set of low dimensional vectors for users and items. Factor models are hard to apply to temporal data. Co-occurrence based models (e.g., markov chain models) [8], [9] are the most popular methods for sequential prediction. Co-occurrence based methods rely on computing the co-occurrence between items in the history. The main limitation about many co-occurrence based methods is the strict assumption about the independence among different items and they merely consider the context information within an extremely short time interval.

In the past few years, deep neural network models have achieved remarkable advance in different tasks in computer vision, speech recognition and natural language processing [10], [11], [12], [13], [5], [6], [7], where the features are learnt in a layer-layer manner. Recurrent Neural Network (RNN) has been successfully applied to sequential data modelling [14], [15], [16], [17], such as word embedding [10], sequential click prediction [18] and image captioning [19]. RNN extended with time-specific transition matrices and distance-specific transition matrices have been used to predict where a user intends to go next [20]. RNN is a suitable tool for modelling temporal information as the hidden representation of RNNs can change dynamically along with the sequential data.

For the query suggestion problem, the previously submitted queries provide valuable context to narrow down ambiguity in the current query and to deliver more focused suggestions [21]. We argue that crowds' map queries have sequential structure at different hierarchical levels of abstraction: a query session is typically composed of a sequence of short-interval queries to accomplish a search mission; a couple of consecutive query sessions depict crowds' geolocalized social-life (i.e., from a hotel to a restaurant), etc. Capturing this hierarchical sequential structure in crowds' map query logs can potentially give more accurate predictions.

To better understand crowds' map query intents, we propose a *Hierarchical Contextual Attention Recurrent Neural*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

Network (HCAR-NN) model in this paper, which attempts to not only learn the *short-range* temporal correlations among queries in a query session, but also capture the *long-range* contextual dependencies among consecutive query sessions. The proposed HCAR-NN consists of three recurrent layers implemented using *Long-Short Term Memory* (LSTM). Given a user's query record, we first partition each user's query record into a couple of short-interval query sessions (i.e., one day or one hour), and we encode each query session into a vector using an *encoding LSTM,* $LSTM^e$. In order to capture the contextual dependencies among query sessions, the encoding output of the previous query session is used to jointly learn the encoding vector of subsequent query session in a soft attention mechanism (*contextual LSTM*, $LSTM^c$). In the end, the *decoding LSTM* ($LSTM^d$) is employed to predict map queries according to the fed encoding vectors.

The main contributions of this paper are summarized as follows:

- We propose a new HCAR-NN model to capture the sequential structures in crowds' query logs (i.e., both the temporal correlations and contextual dependencies) and demonstrate how it can be used to enhance map query suggestions even with the noisy query data.
- We introduce the soft attention mechanism to better capture the *long-range* contextual dependencies among consecutive query sessions and demonstrate how this mechanism can be used to analyze relevant historical query sessions in a dynamic way.
- We train our model over a large, real-world map query dataset. The dataset contains more than 6 million map queries collected from more than 100K different users. Our model can also predict the crowds' search mission with a reasonable accuracy. Moreover, since users' map query logs in some extent depict crowds' geolocalized social life, we visualize the learning vectors of map queries to show the crowd behaviours in a city.

2 Related Work

2.1 Neural Network for Language Processing

Neural networks have been applied to a variety of natural language processing tasks, ranging from information retrieval, language modelling and machine translation.

"Distributed Representation", advocated by Hinton et al. [22], [23], [24] has drawn a lot of attention in recent years. Recently, distributed representation of language has been widely used in natural language processing through learning an embedding vector for each word in an unsupervised learning manner. Wei and Alexander use the artificial neural networks to learn a language model [25]. Toutanova et al. extend the idea of distributed representation of words (words embedding) into the embedding of paragraphs and documents [26]. Bengio et al. employ a neural probabilistic language model to learn a distributed representation for words which allows each training sentence to inform the model about an exponential number of semantically neighbouring sentences [27]. Huang et al. propose a network that learns word embeddings that better capture the semantics of words by incorporating both local and global contexts in documents and at the same time account for homonymy and polysemy by learning multiple embeddings per word [28]. Those learned representations in general capture the semantics of the words and have much more potential than the traditional one-hot-spot word representation.

Mikolov et la. [29] improve the performance of recurrent neural network language models by providing a contextual real-valued input vector, which is used to convey contextual information about the sentence being modelled, in association with each word. Contextual LSTM [17] incorporates contextual features (e.g., topics) into recurrent neural network LSTM model and is employed on word prediction, next sentence selection and sentence topic prediction. Cho et al. [30] and Sutskever et al. [31] use an RNN encoder-decoder architecture to encode a sequence of symbols into a fixed length vector representation, and decode the representation into another sequence of symbols. Li et al. [15] use deep learning models to generate semantic features from in-session contextual information and incorporate these semantic features into the ranking model for clickthrough data. Mitra et al. [32] have studied the distributed representation of queries learnt by a deep neural network model, and shown that it can be used to represent query reformulations as vectors.

2.2 Temporal Prediction

Markov Chain (MC) based methods are a commonly-used approach for temporal prediction. In those methods, the predictions are estimated based on the previous items. Rendle et la. [33] present a method based on personalized transition graphs over underlying Markov chains. In other words for each user a self transition matrix is learned resulting in a transition cube and the cube is factorized with a pairwise interaction model which is a special case of the tucker decomposition.

Nagarajan et al. [34] study the problem of collaborative filtering with interactional context, which is the sequence of clicks made in one session, resulting in personalized and dynamic recommendations to a user engaged in a session. They propose to cluster users according to their transition behaviours (one-step Markov transition probabilities between items) and compute cluster-level Markov models. Jun et al. [35] consider people's forgetting of interest when performing personalized recommendations, and bring forward a personalized framework to integrate interest-forgetting property with Markov model, achieving significant improvement on the accuracy of item recommendation.

The hidden representation of recurrent neural network can change dynamically along with a behavioral history, which makes the recurrent neural network suitable for modeling temporal data. When the records of users' actions are fed into RNN, the parameters in RNN are adaptively tuned to represent users' behavior. Liu et al. [20] extend RNN with spatial and temporal contextual information and use time-specific transition matrices for different time intervals and distance-specific transition matrices for different geographical distances to predict where a user intends to go next.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

2.3 Query Suggestion

Nowadays, query suggestion is a very important capability of on-line search engines. This temporal predicting problem has been extensively studied. Cao et al.[36] propose a context-aware query suggestion approach which first summarises queries into concepts by clustering a click-through bipartite in an off-line learning step, and then constructs a concept sequence suffix tree as the query suggestion model. Baraglia et al. [37] propose an algorithm for updating an existing query flow graph which allows the recommendation model to be kept always updated without reconstructing it from scratch every time, by incrementally efficiently merging the past and present data. Wang et al. [38] propose to rank candidate queries by assessing the joint probability for that the query is selected by the user, that the obtained search results are subsequently clicked by the user, and that the clicked search results ultimately satisfy the users' information need. Sordoni et al. [16] propose a hierarchical recurrent encoder-decoder model that can account for sequences of the previous queries of arbitrary length and avoid data sparsity.

3 Our Approach

Here we give the definitions used in our approach as follows:

- **query record**: The queries submitted by each user in a fixed long-time duration (one month in this paper) are called a query record.
- **query session**: The queries submitted by a user in a short-time interval (one day or half hour) are called a query session. The queries in a query session have strong temporal correlations since they are reformulated to accomplish a user' preferred search mission (e.g., a search for one favorite dinner restaurant).
- **query chunk**: A query chunk consists of several query sessions. The query sessions in a query chunk probably have the contextual dependencies in practice (e.g., a tourist will search a pleasure hotel in the first day and then search a sight spot in the second day). The length of a chunk corresponds with the length of the time interval (number of days, denoted as l in this paper) of that chunk. In this way, if we set l equals to 3, the hierarchical model learns the contextual dependencies among query sessions in 3 days.

Assume that u is one of the users and that his/her query record consists of n query sessions Q_m ($m \in \{1, \cdots, n\}$) in a chronological order, i.e., $Q_m <_t Q_{m+1}$ where $<_t$ is the total order generated by the map query submission time within a duration. A query session Q_m comprises k queries $Q_m = \{q_1^m, q_2^m, \cdots, q_k^m\}$. It should be noted that different query sessions have different numbers of queries. The query record q of user u is denoted as $q = \{\cdots, q_1^m, q_2^m, \cdots, q_k^m, \cdots, q_1^n, q_2^n, \cdots, q_{k'}^n\}$. In the following, we also denote all of queries by user u as $q = \{q_1, q_2, \cdots, q_T\}$ (i.e., the number of queries submitted by user u is T). According to aforementioned definitions, several consecutive query sessions are grouped into a query chunk. In our map query suggestion scenario, one query is usually a name of the location a user is searching for.

For the query records we have collected, we select the queries belonging to four categories (shopping, food, corporation and government) to illustrate the query distribution in one day. That is to say, we calculate the ratio of different categorical queries submitted at each time. Figure 1 illustrates the distribution of the submitted queries belonging to different times in one day (from 00:00 to 23:59). It can be seen that crowd behaviours follow up a *geolocalized* style. For example, people search their preferred lunch restaurants or dinner restaurants from 11:00 am to 12:00 pm or from 17:00 pm to 18:00 pm, respectively.

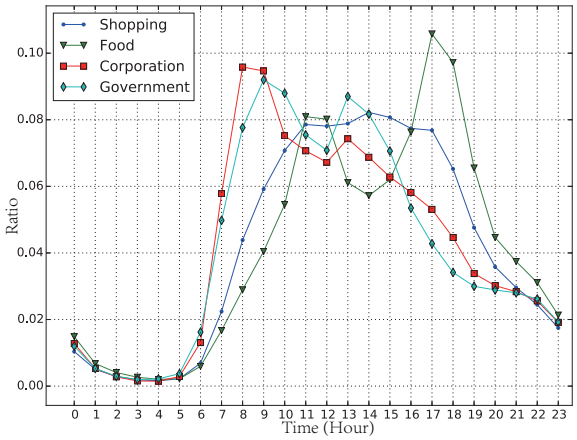


Fig. 1: The illustration of the distribution of submitted queries over time in one day. The queries belong to four categories (shopping, food, corporation and government). It can be seen that crowd behaviors have a geo-social style. For example, people go shopping during the day from 8:00 am to 19:00 pm, and search their preferred lunch restaurants or dinner restaurants from 11:00 am to 12:00 pm or from 17:00 pm to 18:00 pm.

3.1 The Recurrent Neural Network

The *Recurrent Neural Network* (RNN) is widely used to model sequential data. Given the queries by user u, $q = \{q_1, q_2, \cdots, q_T\}$, the standard RNN computes the output sequence of q as $(z_1, z_2, \cdots, z_T)$ by interacting the following equations:

$$\boldsymbol{h}_t = tanh(\boldsymbol{W}_{hx} \cdot \boldsymbol{q}_t + \boldsymbol{W}_{hh} \cdot \boldsymbol{h}_{t-1}) \quad (1)$$

$$\boldsymbol{z}_t = \boldsymbol{W}_{zh} \cdot \boldsymbol{h}_t \quad (2)$$

where $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$, d_h is the number of dimensionality of the hidden output state, $\boldsymbol{q}_t \in \mathbb{R}^d$ is the dense vector of the t^{th} query q_t in q. $\boldsymbol{W}_{hx}$ and $\boldsymbol{W}_{hh}$ are the parameter matrices to be learned. The hidden output state $\boldsymbol{h}_0$ is initialized as a vector filled with 0. The hidden output state at step t is preferred as a summary of the information from time step 0 to time step t.

In order to perform query predictions in a sequence, it is better to learn patterns with wider range temporal dependencies (i.e., the temporal correlations in a query session

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

and the contextual dependencies among query sessions). Here we introduce the Long-Short Term Memory (LSTM) [39] which shows a great success in machine translation [31] and sequence generation [40]. The architecture of LSTM is

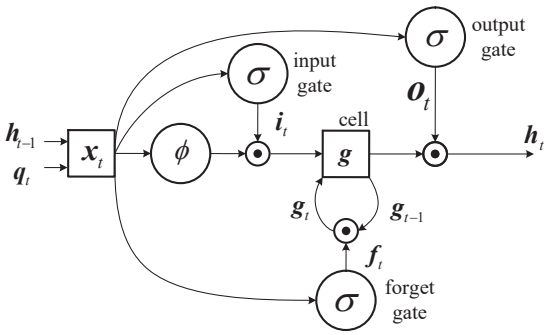


Fig. 2: The architecture of LSTM.

illustrated in Figure 2. There is one memory cell g surrounded by three gates controlling whether to input new data (the input gate i), whether to forget history (the forget gate f), and whether to produce current value (the output gate o) at each time step t. The memory cell in LSTM summarizes the information at each time step of what information has been observed up to now. The value of each gate is calculated according to the input vector $\boldsymbol{q}_t$ at time step t and the hidden output $\boldsymbol{h}_{t-1}$ state at time step $t-1$.

The formal definition of the memory cell and the three gates are given as follows:

$$\boldsymbol{x}_t = [\boldsymbol{q}_t;\ \boldsymbol{h}_{t-1}] \tag{3}$$
$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot \boldsymbol{x}_t + \boldsymbol{b}_i) \tag{4}$$
$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot \boldsymbol{x}_t + \boldsymbol{b}_f) \tag{5}$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot \boldsymbol{x}_t + \boldsymbol{b}_o) \tag{6}$$
$$\boldsymbol{g}_t = \boldsymbol{f}_t \odot \boldsymbol{g}_{t-1} + \boldsymbol{i}_t \odot \phi(\boldsymbol{W}_g \cdot \boldsymbol{x}_t) \tag{7}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{g}_t \tag{8}$$

where $\odot$ represents the element-wise product. σ and ϕ are two element-wise non-linear activation functions. In our experiments, we set σ as sigmoid and ϕ as hyperbolic tangent. $\boldsymbol{W}_i$, $\boldsymbol{W}_f$, $\boldsymbol{W}_o$, and $\boldsymbol{W}_g$ and $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_o$ are the parameter matrices and bias vectors learned during the training.

3.2 The Porposed HCAR-NN Model

Given one user's n query sessions Q_m ($m \in \{1, \cdots, n\}$) as well as the corresponding queries $\{q_1^m, q_2^m, \cdots, q_k^m\}$ in the m^{th} query session Q_m, the hierarchical model we propose captures not only the *short-range* temporal correlations between queries in each query session, but also the *long-range* contextual dependencies between query sessions (i.e., the contextual correlations of a query chunk). In fact, our model consists of three recurrent layers implemented in LSTM: the encoding LSTM ($LSTM^e$), the context LSTM ($LSTM^c$) and the decoding LSTM ($LSTM^d$).

We first input the queries in query session Q_m into $LSTM^e$ and get a feature vector of Q_m. The feature vector of Q_m gives a proper abstraction of its corresponding queries (i.e., the search mission). Since we further attempt to learn the contextual dependencies between query sessions, given a query session Q_m, the latent contextual vector of Q_m is learned by the latent contextual vector of Q_{m-1} and the feature vector of Q_m via the context LSTM ($LSTM^c$). After that, our model computes the log-likelihood of Q_{m+1} using the decoding LSTM ($LSTM^d$) and softmax mapping. In the backward pass, the gradients are calculated and all the parameters are updated.

At the beginning, the embedding vectors of queries (i.e., q_i^m) and query sessions (i.e., Q_m) are initialized randomly and are fine-tuned during the model training. Figure 3 illustrates the overflow of the proposed hierarchical model. Figure 4 depicts the implementation details of our model and Algorithm 1 summarises the training procedure of our model. Table 1 shows the notations we use in our experiments.

TABLE 1: Notations in the HCAR-NN model.

Notation	Description
u	one user
q	one user's query record
n	the number of query sessions in a user's query record
T	the number of queries submitted by one user
L	the set of unique query words in training data
V_L	the size of the query set L
S	one query chunk
Q_m	the m^{th} query session in a query chunk
k	the number of queries in a query session
q_i^m	the i^{th} query in the m^{th} session
$\boldsymbol{q}_i^m$	the embedding vector of query q_i^m
$\boldsymbol{q}_{\#S\#}$	the pre-defined vector of symbol #*START*#
$\boldsymbol{h}_{m,i}^e$	the $LSTM^e$ encoding vector of q_i^m
$\boldsymbol{h}_m^e$	the learned vector of the m^{th} query session
e_t	the attention relevance score
α_t	the attention weight
$\boldsymbol{h}_m^a$	the attention hidden vector for the m^{th} query session
$\boldsymbol{h}_m^c$	the context vector of the m^{th} query session
$\boldsymbol{h}_{m,i}^d$	the $LSTM^d$ decoding vector of q_i^m
$\boldsymbol{p}_i^m$	the probability distribution of q_i^m
θ	all the parameters of our model

3.2.1 Embedding of query sessions

First of all, we respectively embed each query q_i into a latent vector. Given the query set L (the vocabulary of query words), we first represent each query as one-hot representation. The dimensionalities of the one-hot representations equal to the number of queries in the query set L in the training data. After that, the one-hot representations will be transformed into d dimensional vectors as follows:

$$\boldsymbol{q}_i = \eta(q_i) \tag{9}$$

where $\eta(q_i)$ transforms the one-hot representation of query q_i into the latent vector $\boldsymbol{q}_i$ using a learned embedding matrix $\boldsymbol{W}_q$.

Take the m^{th} query session with k queries $Q_m = \{q_1^m, q_2^m, \cdots, q_k^m\}$ in the query chunk S submitted by the user u for example. The query latent vectors $\boldsymbol{q}_1^m, \cdots, \boldsymbol{q}_k^m$ are learned through the query embeddings as Equation (9).

Our encoding LSTM ($LSTM^e$) sequentially takes the query latent vector as input and learns the hidden output vectors as follows:

$$\boldsymbol{h}_{m,i}^e = LSTM^e(\boldsymbol{h}_{m,i-1}^e,\ \boldsymbol{q}_i^m); i = 1, \cdots, k \tag{10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

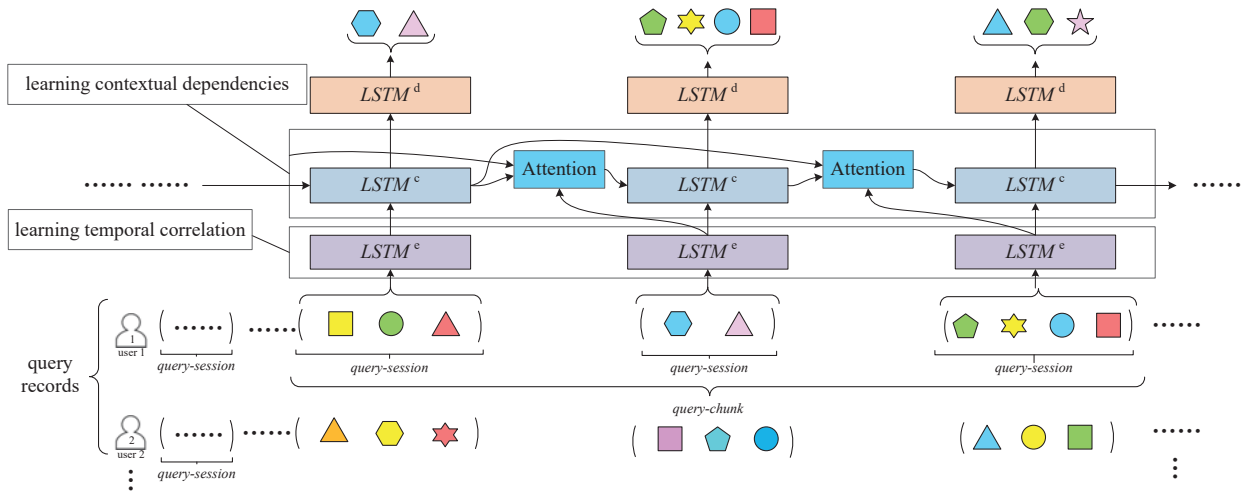


Fig. 3: The intuitive illustration of the overflow of our proposed HCAR-NN. Our model consists of three recurrent layers: the encoding LSTM ($LSTM^e$), the context LSTM with attention mechanism ($LSTM^c$) and the decoding LSTM ($LSTM^d$). Each shape with one color represents a submitted query. Given one user's query record, the query record is split into a sequence of query sessions (the duration of query session is one-day or one-hour according to a specific application). Here three consecutive query sessions are constructed into a query chunk to train the HCAR-NN.

$$\boldsymbol{h}^e_{m,0} = \mathbf{0}; \tag{11}$$

$$\boldsymbol{h}^e_m \equiv \boldsymbol{h}^e_{m,k} \tag{12}$$

where $LSTM^e(\cdot)$ denotes one step forward pass of the encoding LSTM, $\boldsymbol{q}^m_i$ is the latent vector of the i^{th} query in the m^{th} query session, and $\boldsymbol{h}^e_{m,i}$ is the learned hidden output vector of the encoding LSTM of the i^{th} query in the m^{th} query session. The first hidden vector $\boldsymbol{h}_{m,0}$ is a vector filled with the value 0. We set the encoding vector of the m^{th} query session $\boldsymbol{h}^e_m$ to be consistent with the last hidden output vector of the encoding LSTM $\boldsymbol{h}^e_{m,k}$. The encoding vector, which is subsequently fed into the context LSTM, captures the temporal correlations in the query sessions and summarizes the search mission of user u in a short-time interval. The parameters of the encoding LSTM are shared across all of query sessions.

3.2.2 Contextual vector with Attention

It is not easy to capture one user's search intent, since the query log always contains map queries which do not have strong relevance with the user's major search intent (e.g., a tourist may submit one query of toilet after searching some tourist attractions, and then continues to query other tourist attractions, or someone who keeps searching grocery stores nearby may be interrupted by his friend who wants to search a hospital). We call the queries which are not strongly relevant to the user's major search intent as *noised queries*. To give a better map query suggestion, we integrate the attention mechanism which can learn the context vector from the representations of relevant historical query sessions in a dynamic way. The soft attention mechanism [41] has achieved a great success in several sequence modelling tasks. The major concern of the soft attention is to generate a new sequence, using the dynamically learned attention weights from the original sequence.

Given the encoding vector $\boldsymbol{h}^e_m$ and its learned previous sessions' context vectors $(\boldsymbol{h}^c_{m-1}, \boldsymbol{h}^c_{m-2}, \cdots, \boldsymbol{h}^c_{m-r})$ (here r is the *attention range* which indicates how many previous sessions we choose to focus on), we first measure the relevance between the current query session and its previous query sessions by calculating the relevance score as follows:

$$e_t = \boldsymbol{\omega}_a \tanh(\boldsymbol{W}_a \boldsymbol{h}^e_m + \boldsymbol{U}_a \boldsymbol{h}^c_{m-t} + b_a) \tag{13}$$

where $\boldsymbol{\omega}_a$, $\boldsymbol{W}_a$, $\boldsymbol{U}_a$ and b_a are all parameters to be learned. e_t is the real value relevance score. For each previous query session $\boldsymbol{h}^c_{m-t}$, the corresponding attention weight α_t is calculated according to its relevance score as follows:

$$\alpha_t = \exp(e_t) / \sum_{j=1}^{r} \exp(e_j) \tag{14}$$

We calculate the attention hidden vector by summing up the previous session context vectors according to their attention weights as follows:

$$\boldsymbol{h}^a_m = \sum_{t=1}^{r} \alpha_t \boldsymbol{h}^c_{m-t} \tag{15}$$

The context LSTM takes as input the encoding vector $\boldsymbol{h}^e_m$ and the attention hidden vector learned through the soft attention mechanism, and computes a new context vector $\boldsymbol{h}^c_m$ as follows:

$$\boldsymbol{h}^c_m = LSTM^c(\boldsymbol{h}^a_m, \boldsymbol{h}^e_m) \tag{16}$$

where $LSTM^c(\cdot)$ denotes the LSTM forward pass for learning the context vector. In Algorithm 1, $Attention(\cdot)$ indicates the process of computing the attention hidden vector.

The context LSTM captures the contextual dependencies and learns the user's query patterns by going through his/her previous query sessions. The soft attention mechanism could make the context LSTM to focus on a subset of the previous query sessions and reduce the misleading effect of some noised query sessions. Each $\boldsymbol{h}^c$ bears a particularly powerful characteristic: it is somehow sensitive to the query order and can potentially encode order-dependent reformulation patterns (i.e., the contextual dependencies) [42].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

3.2.3 Predicting Next queries

Decoding LSTM ($LSTM^d$) aims to figure out what queries are submitted in the next according to the captured user's previous queries. The decoding LSTM decodes the learned context vector into new queries as follows:

$$\begin{aligned} \boldsymbol{h}^d_{m+1,i} &= LSTM^d(\boldsymbol{h}^d_{m+1,i-1},\ \boldsymbol{h}^c_m,\ \boldsymbol{q}^{m+1}_{i-1}) \\ \boldsymbol{h}^d_{m,0} &= \boldsymbol{0}, \boldsymbol{q}^m_0 = \boldsymbol{q}_{\#S\#}; \\ m &= 1, \cdots, n;\ i = 1, \cdots, k^{m+1} \end{aligned} \tag{17}$$

where $\boldsymbol{h}^d_{m+1,i-1}$ is the decoding vector computed before time step i, $\boldsymbol{h}^c_m$ is the context vector of the m^{th} query session, and $\boldsymbol{q}^{m+1}_{i-1}$ is the $(i-1)^{th}$ latent query vector in the $(m+1)^{th}$ query session. The starting vector $\boldsymbol{q}_{\#S\#}$ is the latent vector of a pre-defined query symbol *#START#*. k^{m+1} indicates the number of queries in the $(m+1)^{th}$ query session. The decoding vectors $\boldsymbol{h}^d_{m+1,1}, \cdots, \boldsymbol{h}^d_{m+1,k^{m+1}}$ are then be used to calculate the probability of new queries. The parameters of the decoding LSTM are shared across the query session.

To calculate the probability distribution of new queries, we use a softmax mapping layer. The mapping layer projects the decoding vectors into vectors of dimension V_L (the size of the query set L). Then the query with the highest probability is considered as the predicted query at each time step.

$$\boldsymbol{p}^m_i = softmax(\boldsymbol{W}_d \cdot \boldsymbol{h}^d_{m,i} + \boldsymbol{b}_d) \tag{18}$$

where $\boldsymbol{p}^m_i$ is the probability distribution of the i^{th} query in the m^{th} query session, $\boldsymbol{h}^d_{m,i}$ is the decoding vector, $\boldsymbol{W}_d$ is the mapping matrix and $\boldsymbol{b}_d$ is the bias vector.

The objective function to optimize is the log-likelihood over the whole training data as follows:

$$\max_{\theta} \sum_m \sum_i logPr(q^m_i | q^m_{i-1}, \cdots q^m_1, Q_{m-1}, \cdots, Q_1; \theta) \tag{19}$$

where q^m_i is the ground-truth query. $q^m_{i-1}, \cdots q^m_1$ indicate the queries before q^m_i in the query session Q_m, $\{Q_{m-1}, \cdots, Q_1\}$ indicates the query sessions before the m^{th} query session and θ represents all the model's parameters.

4 Experiments

4.1 Dataset

The dataset we use to evaluate our model is collected from the *Baidu Map* on-line services. This dataset is collected from 118,468 users in the capital city of China *Beijing*. It consists of 6,590,157 queries, covering 21,358 locations (e.g., Palace Museum and Peking Union Medical College Hospital), submitted in one month (the December of 2015). Each user is represented as a unique *user-id* in the on-line service system. The queries submitted by one user in one day are taken as one query session. In total, we have 2,362,072 query sessions and use 1,889,657 query sessions (about 80 %) for training, 236,207 query sessions for validation and 236,207 query sessions for testing. The statistics information of our dataset is summarized in Table 2. While in our experiments, we regard each query chunk as a data sample for training or validation or testing. A query chunk consists of several query sessions. If a query chunk is split into the training set, all of its query sessions belong to the training set.

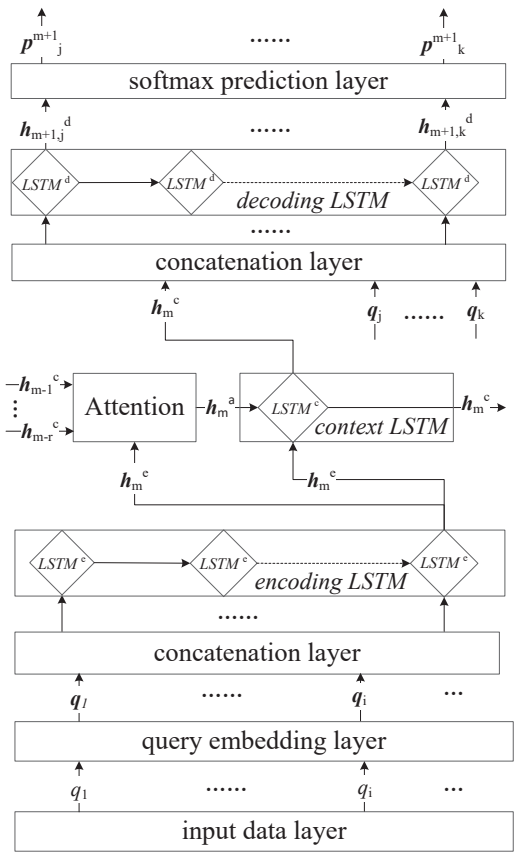


Fig. 4: The implementation details of HCAR-NN model.

4.2 Comparison Methods

We compare our model with several prediction methods:

- **MF**: The matrix factor model [43] is one of the state-of-the-art methods for conventional collaborative filtering.
- **MC**: The Markov chain model is a classical sequential model [20]. Here we use Markov chain model to learn the temporal correlations between two neighboring queries. That is to say, only two neighboring two queries are used to train MC for map suggestion.
- **Flat-LSTM**: We use each query session (i.e., the queries in one day) to train LSTM for map suggestion and call it *Flat-LSTM*. In Flat-LSTM, the temporal correlations between queries in each query session are learned. However, the contextual dependencies among query sessions are ignored in *Flat-LSTM*.
- **HCR-NN(chunk=*l*)**: The simple version of our proposed hierarchical model without attention mechanism. For the m^{th} query session, the context LSTM of HRC-NN takes as input the encoding vector $\boldsymbol{h}^e_m$ and the context vector learned from the previous query session $\boldsymbol{h}^c_{m-1}$, and computes the new context vector

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1: Algorithm for training HCAR-NN model

Input: One batch $\mathcal{B}$ of training data, containing many query chunks.

foreach *query chunk* $S \in \mathcal{B}$ **do**
  n = number of query sessions in the query chunk S.
  k^m = number of queries in the m^{th} query session.
  Embedding the queries into latent vectors $\boldsymbol{q}_1^1, \cdots, \boldsymbol{q}_{k^m}^m$.
  for $m \leftarrow 1$ **to** $n-1$ **do**
    /* Forward Passing */
    /* Encoding the query session */
    for $i \leftarrow 1$ **to** k^m **do**
      $\boldsymbol{h}_{m,i}^e = LSTM^e(\boldsymbol{h}_{m,i-1}^e, \boldsymbol{q}_i^m)$;
    end
    $\boldsymbol{h}_m^e \equiv \boldsymbol{h}_{m,k^m}^e$
    /* Learning the context vector */
    $\boldsymbol{h}_m^a = Attention(\boldsymbol{h}_m^e, \boldsymbol{h}_{m-1}^c, \cdots, \boldsymbol{h}_{m-r}^c)$
    $\boldsymbol{h}_m^c = LSTM^c(\boldsymbol{h}_m^a, \boldsymbol{h}_m^e)$
    /* Decoding the next queries */
    for $i \leftarrow 1$ **to** k^{m+1} **do**
      $\boldsymbol{h}_{m+1,i}^d = LSTM^d(\boldsymbol{h}_{m+1,i-1}^d, \boldsymbol{h}_m^c, \boldsymbol{q}_{i-1}^{m+1})$
      $\boldsymbol{p}_i^{m+1} = softmax(\boldsymbol{W}_d \cdot \boldsymbol{h}_{m+1,i}^d + \boldsymbol{b}_d)$
    end
    /* Backward Passing */
    Calculate and accumulate the gradients.
  end
end
Calculate the update values $\nabla\theta$
/* Update the parameters */
$\theta = \theta - \nabla\theta$

Output: The parameters of the model θ

TABLE 2: The statistics information of the dataset.

query	No. queries	6,590,157
	No. query sessions	2,362,072
	No. different map query words	21,358
No.user	total	118,468
	No. query sessions ≤10	2,830
	No. query sessions >10 & ≤20	64,679
	No. query sessions >20	50,959

$\boldsymbol{h}_m^c$ as follows:

$$\boldsymbol{h}_m^c = LSTM^c(\boldsymbol{h}_{m-1}^c,\ \boldsymbol{h}_m^e);\ \boldsymbol{h}_0^c = \mathbf{0} \tag{20}$$

where $LSTM^c(\cdot)$ denotes the LSTM forward pass for learning the context vector. The default context vector $\boldsymbol{h}_0^c$ is a vector filled with 0.
The model can be trained using different lengths of query chunks l. The length of a chunk corresponds with the length of the time interval (number of days, denoted as l) of that chunk. In this way, if we set l equals to 3, the hierarchical model learns the contextual dependencies among query sessions in 3 days.

- **HCAR-NN(chunk=*l*)**: Our proposed hierarchical model HCAR-NN can be trained using different sizes of query chunks l. In the experiments, we show the performance of our approach in terms of different sizes of query chunks.

4.3 Evaluation metrics

For prediction tasks, the classical metrics *Recall-at-K* (R@K) [44] and *Mean-Reciprocal-Rank* (MRR) [45] are widely used to evaluate the performance of different algorithms.

- **Recall@K**: We use *Recall-at-K* to measure whether the correct predictions are ranked ahead of others. We use Recall@K (K=1,5,10,15,20) to compute the fraction of the times where the correct prediction is ranked among the top K predicted items. Given N data samples, according to the corresponding R predicted items, Recall@K is defined as follows:

$$R@K = \frac{1}{N}\sum_i^N \epsilon(r_i \leq K) \tag{21}$$

where N is the number of the data samples and r_i is the rank value of the correct prediction for the sample i. $\epsilon(r_i \leq K) = 1$ if rank value of the correct prediction is less than or equal K, otherwise $\epsilon(r_i \leq K) = 0$.
- **MRR**: *Mean Reciprocal Rank* is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. The reciprocal rank of a prediction is the multiplicative inverse of the rank. The *Mean-Reciprocal-Rank* is the average of the reciprocal ranks of predictions, as follows:

$$MRR = \frac{1}{N}\sum_i^N \frac{1}{r_i} \tag{22}$$

where r_i is the rank position of the correct prediction for sample i. N is the number of the data samples.

4.4 Experimental Results

Here we report the comparisons in two tasks: *Query Prediction* and *Robustness*. We also evaluate our model on *Prediction of Search Mission* and show the learned representations of queries in an intuitive way.

4.4.1 Map Query Prediction

Given the query history submitted by one user, we intend to predict the queries that will be submitted by the same user next time. Table 3 shows the performance comparisons of query prediction. In Figure 5 we illustrate the prediction accuracy in terms of Recall@1 of different categories. Table 4 shows five examples of the sequence of actual queries input by users and the corresponding candidate queries predicted from our model. We make the following observations:

- The temporal correlations between map queries are really beneficial to understanding crowds' query intent in a short-time interval. We observe that the approach MF which does not capture the temporal correlations among queries is not as effective as other context sensitive counterparts.
- The RNN based methods (Flat-LSTM, HCR-NN and HCAR-NN) which effectively encode a wider context

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

of map queries (i.e., the context of query sessions) than the Markov chain model that only captures the narrow context between neighbouring queries are more effective for map query suggestion.
- By hierarchically capturing the temporal correlations and contextual dependencies in query records, our proposed HCAR-NN as well as HCR-NN model outperforms the Flat-LSTM method.
- By dynamically capturing the *long-range* contextual dependencies using soft attention mechanism, our proposed HCAR-NN outperforms the hierarchical model without attention mechanism HCR-NN.
- The length of query chunk has an influence on the performance of our model. If we set the size of query chunk as a larger value (e.g., 5), the performance is deteriorated. A possible reason is that crowds' geo-social life in general keeps a short-time tendency style.
- Our proposed HCAR-NN model understands users' query intent. As shown in Table 4, our model understands the users' intent for a district in example 1, intent for a market in example 2, intent for a restaurant in example 3, intent for a hotel in example 4 and intent for a gas station in example 5. Our model always generates candidate queries which are relevant with the ground truth query.

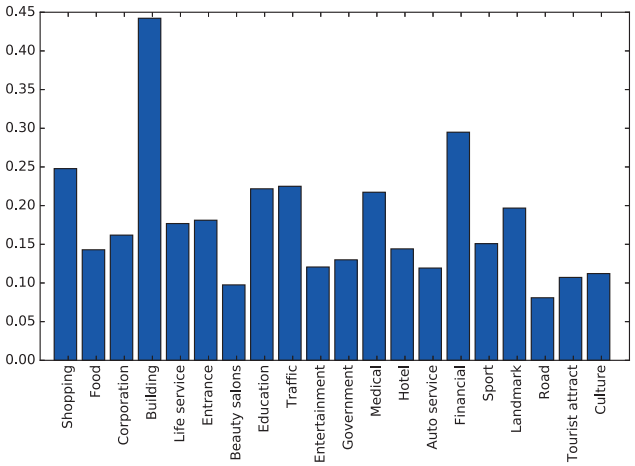


Fig. 5: The prediction accuracy of different query categories in terms of Recall@1.

4.4.2 Robustness of our Model

In this section we evaluate the robustness of our model on some hand crafted datasets. We create the query-missed dataset by removing some queries from the original dataset randomly and generate the noised dataset by inserting some noised query sessions into the original dataset.

To evaluate the generalization capability of our model on the situation when some queries are missing, we make the following changes to the map query dataset and evaluate the performances on different conditions:

- **Missing-The-First**: The first query of each query session is removed (i.e., missing).
- **Missing-The-Second**: The second query of each query session is removed.
- **Missing-Random**: We randomly drop one query in each query session in the training data, and evaluate the performance of different models on this new incomplete dataset.
- **Missing-Two**: Two neighbouring queries of each query session are removed.

To evaluate the robustness of our model on noised dataset, we first randomly select about 6000 unnoised data samples, and then insert one randomly noised query session after every 2, 3 and 4 original query sessions. The noised query sessions are the query sessions which are not coincident with the previous query sessions in terms of category. Thus we evaluate the performance on the following conditions:

- **Noised 1/3**: One noised query session is inserted after every 2 query sessions, so the rate of noised session in this dataset is 1/3.
- **Noised 1/4**: One noised query session is inserted after every 3 query sessions, so the rate of noised session in this dataset is 1/4
- **Noised 1/5**: One noised query session is inserted after every 4 query sessions, so the rate of noised session in this dataset is 1/5.

In Table 5, our proposed model always achieves the best performance in the query-missed cases and the noised cases, demonstrating the power of hierarchical abstraction of query records of this model.

4.4.3 Inference of session length

Besides the length of query chunk, we want to examine the impact of the length for the short-term dependencies (query session). In this section, we evaluate our model on different settings of session length from 10 minutes to 12 hours. In this experiment, we fix the length of query chunks equal to 3. Table 6 gives us the results of our model while changing the session length from 10 minutes to 12 hours. **R@N** is short for **Recall@N** and 'm' is short for 'minute', 'h' is short for 'hour'.

TABLE 6: The performance (in percentage) of our model on the case of different session lengths.

Metrics	10 m	30 m	1 h	3 h	6 h	12 h
R@1	9.915	9.989	10.311	**10.311**	9.961	9.739
R@5	17.683	17.468	18.031	**18.124**	17.753	17.869
R@10	20.566	20.334	20.995	**21.316**	20.959	21.232
R@15	22.318	22.165	22.852	**23.325**	23.003	23.187
R@20	23.639	23.535	24.263	**24.859**	24.523	24.761
MRR	13.818	13.799	14.233	**14.305**	13.952	13.863

The results in Table 6 show that 3h is the best length for query sessions. A possible reason is that people's search intent in general consists in hours.

4.4.4 Inference of attention range

By adopting the soft attention mechanism, the query context can be learned and captured better in a dynamic way. To examine the impact of the attention range which indicates how

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

TABLE 3: The performance (in percentage) comparisons of query prediction. The results shown in boldface are the best results.

Models	Recall@1	Recall@5	Recall@10	Recall@15	Recall@20	MRR
MF	3.468	5.590	6.628	7.329	7.886	4.665
MC	5.632	10.510	12.626	13.956	15.006	8.185
Flat-LSTM	7.095	12.053	14.373	15.920	17.120	9.753
HCR-NN(chunk=1)	7.842	13.898	16.738	18.653	20.113	11.070
HCR-NN(chunk=2)	8.276	14.849	17.765	19.623	21.069	11.709
HCR-NN(chunk=3)	8.597	15.316	18.196	20.100	21.542	12.092
HCR-NN(chunk=4)	8.529	15.069	17.875	19.683	21.078	11.927
HCR-NN(chunk=5)	8.141	14.323	17.101	18.901	20.258	11.372
HCAR-NN(chunk=3)	**9.855**	**17.563**	**20.738**	**22.732**	**24.256**	**13.804**
HCAR-NN(chunk=4)	9.678	17.370	20.540	22.496	23.971	13.601
HCAR-NN(chunk=5)	9.663	17.063	20.219	22.179	23.670	13.470

TABLE 4: Examples of the sequences of actual queries input by users and the corresponding queries generated from our model

	User Input Examples	Ground Truth Query	Top Candidates
1	Zuojia Village → No.379 Bus Stop → Building Materials Market → ?	Yuxin District	Yuxin District, Zuojia Village, Guomen Building
2	Liuli Bridge → Beijing Electric Hospital → Wumart Supermarket → ?	MerryMart	WuMart, MerryMart Supermarket, MerryMart
3	Chaoqinghui → Nanxincang Building → KFC → ?	McDonald's	McDonald's, KFC, Starbucks
4	Fengtai Technology Park → 7 Days Inn → ?	Hai You Hotel	Home Inns, Hai You Hotel, Fengtai South Road
5	PetroChina → Sinopec → ?	PetroChina Gas Station	PetroChina, PetroChina Gas Station, Sinopec Gas Station

many previous query sessions we choose to focus on, we evaluate our model on different settings of attention range from 3 to 11. Table 7 gives the results of our model while changing the attention range from 3 to 11. The attention is computed within a chunk. Suppose a chunk with length 3 (the chunk consists of all the query sessions in consecutive 3 days), and the length of a query session is 3 hours. The number of query sessions in this chunk is no more than 24 (3*24/3) in the chunk. If the attention range is 7, then while our model is learning a query session in the chunk, it will pay attention to at most 7 previous query sessions. In this experiment, we fix the session length as 30 minutes and the chunk length as 3. As show in Table 7, the range of attention should not be too large (as the relevance may be decrease rapidly in temporal), nor too small (as the context LSTM need rich information to learn the hidden vector).

Figure 6 illustrates the distribution of attention weights as the attention range r changes from 3 to 10. The x-axis denotes the relative position between current session and its neighboring. Suppose the position of current query session is m, we use $m-3$ to denote the query session submitted the third times before. The y-axis denotes the average values of attention weights how each position influences the query session in position m. The dash line denotes the position of value $\frac{1}{r}$. Our model tends to pay more attention to neighboring context sessions as they may be much more relevant to users' current query intent. As illustrated in the Figure 6, neighboring context sessions always gain higher attention weights than those context sessions which are far from current query.

4.4.5 Attention Weights Analysis

In our proposed HCAR-NN model, the soft attention mechanism can learn the attention weights for previous query sessions according to their relevance to the next query session. In Figure 7 we illustrate the attention weights learned by our model while predicting the next query. In Figure 7 the height of each bar equals to the weight learned for that query session and the category each query belongs to is also given. The bar of each noised query (i.e., whose belonging category is not consistent with those of its neighboring queries) is denoted with oblique line and the category name of each noised query is denoted with underline. The queries in each example are arranged in a temporal order and the category names at the right side of each arrow are the category names of the next ground truth query. From Figure 7, we can see that historical queries that belong to the same category as that of the next ground truth query are assigned with high weights, while queries that belong to a different category are assigned with low weights by our model.

4.4.6 Noised and Unnoised Query Chunks

In this section we evaluate our model on noised query chunks and unnoised query chunks respectively. We randomly select about 3000 noised query chunks and about

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

TABLE 5: The performance (in percentage) comparisons of query prediction when some queries in each query session are removed. The results shown in boldface are the best results.

Models	Recall@1	Recall@5	Recall@10	Recall@15	Recall@20	MRR
Missing-The-First						
MF	1.652	3.236	4.229	5.003	5.617	2.661
MC	5.458	10.034	11.958	13.217	14.226	7.864
Flat-LSTM	6.500	11.882	14.456	16.146	17.419	9.377
HCR-NN(chunk=3)	7.336	13.697	16.582	18.428	19.816	10.637
HCAR-NN(chunk=3)	**7.376**	**13.814**	**16.719**	**18.563**	**19.972**	**10.719**
Missing-The-Second						
MF	1.610	3.114	4.087	4.824	5.442	2.581
MC	5.387	10.010	11.865	13.067	14.068	7.796
Flat-LSTM	5.684	11.044	13.566	15.252	16.558	8.542
HCR-NN(chunk=3)	6.574	12.520	15.257	17.045	18.414	9.711
HCAR-NN(chunk=3)	**7.016**	**13.465**	**16.370**	**18.215**	**19.623**	**10.378**
Missing-Random						
MF	1.548	2.985	3.976	4.697	5.330	2.501
MC	5.482	10.074	11.948	13.211	14.244	7.894
Flat-LSTM	6.201	12.039	14.710	16.402	17.705	9.277
HCR-NN(chunk=3)	6.481	12.756	15.602	17.426	18.827	9.765
HCAR-NN(chunk=3)	**7.082**	**13.738**	**16.659**	**18.542**	**19.965**	**10.527**
Missing-Two						
MF	1.345	2.669	3.476	4.072	4.571	2.166
MC	5.218	9.550	11.303	12.568	13.578	7.514
Flat-LSTM	5.340	9.951	12.153	13.577	14.705	7.855
HCR-NN(chunk=3)	5.563	10.591	13.042	14.623	15.879	8.239
HCAR-NN(chunk=3)	**5.679**	**10.682**	**13.144**	**14.777**	**16.037**	**8.355**
Noised 1/3						
MF	4.804	7.909	8.639	9.034	9.315	6.263
Flat-LSTM	4.450	8.207	10.059	11.214	12.121	6.470
HCR-NN(chunk=3)	12.114	18.562	20.781	22.181	23.265	15.429
HCAR-NN(chunk=3)	**12.607**	**21.404**	**24.120**	**25.772**	**27.004**	**16.955**
Noised 1/4						
MF	5.762	9.481	10.356	10.826	11.161	7.504
Flat-LSTM	5.068	9.344	11.449	12.762	13.791	7.361
HCR-NN(chunk=3)	13.819	21.119	23.707	25.317	26.529	17.600
HCAR-NN(chunk=3)	**14.355**	**24.440**	**27.486**	**29.371**	**30.785**	**19.313**
Noised 1/5						
MF	6.105	10.065	10.972	11.470	11.821	7.949
Flat-LSTM	5.272	9.719	11.910	13.274	14.343	7.657
HCR-NN(chunk=3)	14.385	22.067	24.684	26.361	27.621	18.318
HCAR-NN(chunk=3)	**14.946**	**25.397**	**28.613**	**30.569**	**32.036**	**20.105**

TABLE 7: The performance (in percentage) of our model on the case of different attention ranges.

Metrics	3	4	5	6	7	8	9	10	11
Recall@1	9.989	10.081	10.000	10.228	10.231	10.213	9.956	**10.266**	10.139
Recall@5	17.468	17.585	17.590	17.949	**18.404**	18.272	17.239	17.429	17.894
Recall@10	20.334	20.496	20.500	20.884	**21.395**	21.253	20.111	20.243	20.899
Recall@15	22.165	22.376	22.297	22.772	**23.242**	23.108	21.918	22.074	22.786
Recall@20	23.535	23.792	23.671	24.176	**24.629**	24.485	23.320	23.443	24.191
MRR	13.799	13.908	13.853	14.143	**14.326**	14.264	13.687	13.948	14.080

3000 unnoised query chunks from the test data split. Then we evaluate the performances of different models on the noised and unnoised query chunks in terms of map query prediction. Table 8 shows the performances of different models. From Table 8 we see that it is much harder to correctly predict the next map query with noised query histories than unnoised query histories and the model we have proposed always has the better performance than other models.

4.4.7 Prediction of Search Mission

Typically a user completes his/her search mission via a sequence of relevant reformulated queries and attempts to clarify his/her need within a short-time interval. For example, a user is planning a dinner with friends and starts issuing a few relevant queries, such as nearby Chinese restaurants. In this scenario, the submitted queries belong to the same category (i.e., food). In our dataset, most of the queries are given an appropriate category such as Food

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

TABLE 8: The performance (in percentage) comparisons of query prediction on noised query chunks and un-noised query chunks. The results shown in boldface are the best results.

Models	Recall@1	Recall@5	Recall@10	Recall@15	Recall@20	MRR
Noised Query Chunks						
MF	3.185	4.873	5.466	5.805	6.095	4.040
MC	3.592	7.955	9.971	11.178	12.870	5.934
Flat-LSTM	4.542	9.068	11.473	13.006	14.293	7.031
HCR-NN(chunk=3)	6.695	12.744	15.805	17.653	18.897	9.892
HCAR-NN(chunk=3)	**6.947**	**13.799**	**16.876**	**18.908**	**20.368**	**10.509**
Unnoised Query Chunks						
MF	4.116	5.910	6.679	7.176	7.528	5.090
MC	4.252	8.035	9.767	11.107	12.082	6.303
Flat-LSTM	5.291	10.036	12.383	14.127	15.573	7.868
HCR-NN(chunk=3)	9.356	15.166	17.917	19.649	21.059	12.473
HCAR-NN(chunk=3)	**9.767**	**16.186**	**18.911**	**20.636**	**21.964**	**13.157**

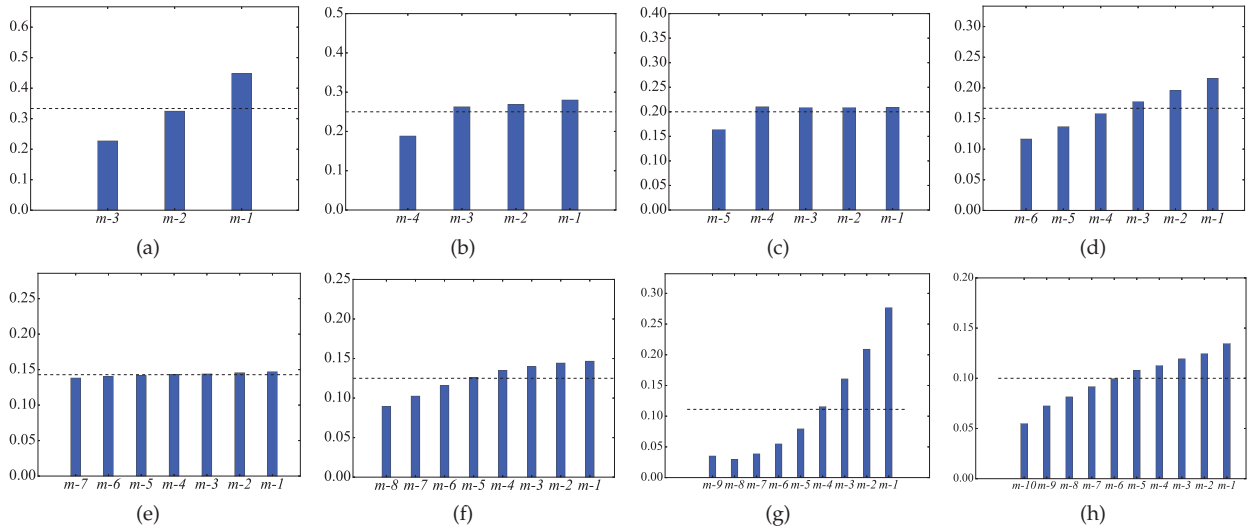


Fig. 6: The distribution of attention weights as attention range changes. The x-axis denotes the relative position between current session and its neighboring. Suppose the position of current query session is m. The y-axis denotes the average values of attention weights how each position influences the query session in position m. The dash line denotes the position of value $\frac{1}{r}$ and r is the value of the attention range.

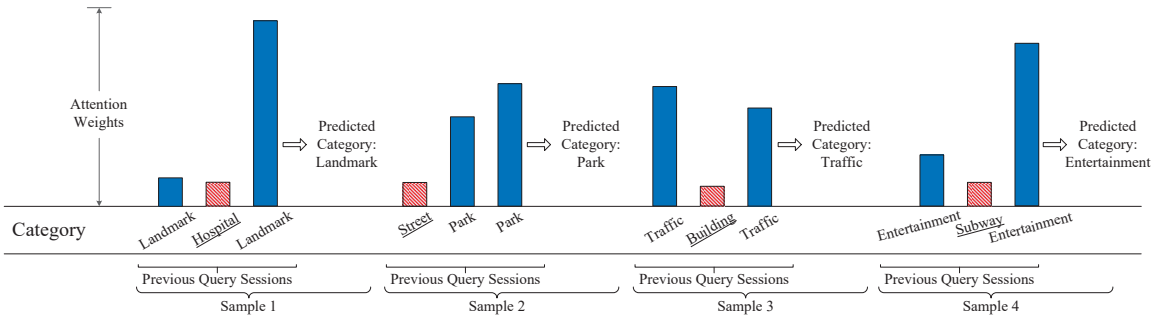


Fig. 7: The attention weights learned by our model while predicting the next query. The height of each bar equals to the weight learned for that query session and the category each query belongs to is also given. The bar of each noised query (i.e., whose belonging category is not consistent with that of its neighbouring queries) is denoted with an oblique line and the category name of each noised query is denoted with underline. The queries in each example are arranged in a temporal order and the category names at the right side of each arrow are the category names of the next ground truth query.

or Park. Given a query session, we call the query session *mission-consistent* if all the queries in the session belong to the same category. After we obtain the predicted query sessions, we remove those predicted query sessions with

their ground truth as not mission-consistent. For all the remaining mission-consistent sessions, we define the *predicted consist rate* of searching sessions to evaluate the prediction performance of mission consistent query sessions as follows:

$$r^{p}_{consistent} = \frac{No.\ predicted\ consistent\ sessions\ w.r.t.\ a\ category}{No.\ mission\ consistent\ sessions\ w.r.t.\ a\ category} \quad (23)$$

In the calculation of the predicted consistent rate, we only consider whether the top ranked predicted query belongs to the category consistent with the ground truth.

Table 9 shows the predicted consistent rate of the search missions in terms of several categories. The higher the predicted consistent rates are, the clearer the query sessions pertain to a search mission. In Table 9, we also give out the *ground truth consistent rate* of query sessions belonging to the corresponding categories defined as follows:

$$r^{g}_{consistent} = \frac{No.\ mission\ consistent\ sessions\ w.r.t.\ a\ category}{No.\ mission\ consistent\ sessions\ w.r.t.\ all\ ategories} \quad (24)$$

From Table 9, we see that queries in many of the query sessions are really relevant to an explicit mission (i.e., Building, Landmark, Food and Shopping); many query sessions do not deliver a direct search mission (i.e., Auto service and Beauty salons). It is also clear that our HCAR-NN does demonstrate a reasonable performance of predicting search mission in Table 9.

Since users accomplish their search mission within a very short-time interval, during the calculation of mission consistence, the query session consists of queries one user submitted within a half-hour duration.

4.4.8 Embedding of Queries

After training our model, we have learned a dense vector representation (query embedding) for each query in the training data. Those embeddings not only are compact in memory, but also capture the semantics of the words of queries. In Figure 8, we use t-SNE [46] to project the learned query embeddings into 2 dimensional space. As illustrated in Figure 8 the embedding space captures the semantic similarity between map queries. That is to say, queries belonging to categories (i.e., bank, hospital, park, and hotel) keep their context (neighborhood similarity in their vicinity) in the learned embedding space.

4.4.9 Time Complexity Analysis

The time complexity of the forward pass of one LSTM unit is $(d_q + d_h) \times d_h$ where d_q and d_h are the dimensionalities of the latent embedding vector of the query and the hidden output vector of the LSTM unit, respectively. Here we set all the dimensionalities of hidden vectors equal to d. Thus the time complexity of one LSTM unit is $O(d^2)$. Suppose a query session contains at most k consecutive queries; the time complexity of the forward pass of this query session through our model is $O(kd^2) + O(d^2) + O(kd^2) \in O(kd^2)$. Given c query chunks, for each query chunk there are at most n consecutive query sessions; the forward pass time complexity of those query chunks is $O(cnkd^2)$.

5 Conclusion

In this paper, we have proposed a new model to predicting map queries in an encoding-decoding manner. The proposed model, called *Hierarchical Contextual Attention Recurrent Neural Network* (HCAR-NN), not only learns the temporal correlations among map queries in a query session (queries submitted by a user in a short-time interval), but also captures the contextual dependencies among map query sessions. To better capture the *long-range* contextual dependencies, we introduce the soft attention mechanism. The experimental evaluations show the performance improvements of our method over the state-of-the-art methods. In addition to map query suggestions, our model is general enough to be used in a variety of other applications.

References

[1] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola, "Maximum margin matrix factorization for collaborative ranking," *Advances in neural information processing systems*, pp. 1–8, 2007.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[3] B. Hidasi and D. Tikk, "Fast als-based tensor factorization for context-aware recommendation from implicit feedback," in *Proceedings of the 2012 European conference on Machine Learning and Knowledge Discovery in Databases-Volume Part II*. Springer-Verlag, 2012, pp. 67–82.

[4] R. S. Zhou Zhao, X. H. Xing Xie, and Y. Zhuang, "Mobile query recommendation via tensor function learning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI '15, 2015.

[5] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1029–1038.

[6] J. Song, S. Tang, J. Xiao, F. Wu, and Z. M. Zhang, "Lstm-in-lstm for generating long descriptions of images," *Computational Visual Media*, vol. 2, no. 4, pp. 379–388, 2016.

[7] D. Kong, F. Wu, S. Tang, and Y. Zhuang, "Ad recommendation for sponsored search engine via composite long-short term memory," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 416–420.

[8] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," ser. IJCAI '13. AAAI Press, 2013, pp. 2605–2611.

[9] M. Chen, Y. Liu, and X. Yu, "Nlpmm: a next location predictor with markov modeling," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2014, pp. 186–197.

[10] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *INTERSPEECH*, vol. 2, 2010, p. 3.

[11] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, "Rnnlm-recurrent neural network language modeling toolkit," in *Proc. of the 2011 ASRU Workshop*, 2011, pp. 196–201.

[12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] S. Lawrence, C. L. Giles, and S. Fong, "Natural language grammatical inference with recurrent neural networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 12, no. 1, pp. 126–140, 2000.

[15] X. Li, C. Guo, W. Chu, Y.-Y. Wang, and J. Shavlik, "Deep learning powered in-session contextual ranking using clickthrough data," in *Proc. of NIPS*, 2014.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2700392, IEEE Transactions on Knowledge and Data Engineering

TABLE 9: The performance of predicting search missions of different query sessions belonging to different categories as well as the consistent rates of query sessions in ground-truth dataset.

Category	$r^p_{consistent}$ (%)	$r^g_{consistent}$ (%)	Category	$r^p_{consistent}$ (%)	$r^g_{consistent}$ (%)
Shopping	18.15	8.77	Food	13.28	7.16
Corporation	6.69	1.61	Building	21.88	31.26
Life service	11.41	2.55	Entrance	8.94	0.76
Beauty salons	9.17	0.19	Education	15.15	4.30
Traffic	25.31	6.96	Entertainment	10.51	2.22
Government	7.59	1.72	Medical	9.51	3.20
Hotel	11.89	5.03	Auto service	5.56	0.64
Financial	16.27	3.09	Sport	12.23	0.95
Landmark	20.09	10.05	Road	8.30	2.27
Tourist attractions	9.68	2.66	Culture	8.98	0.56

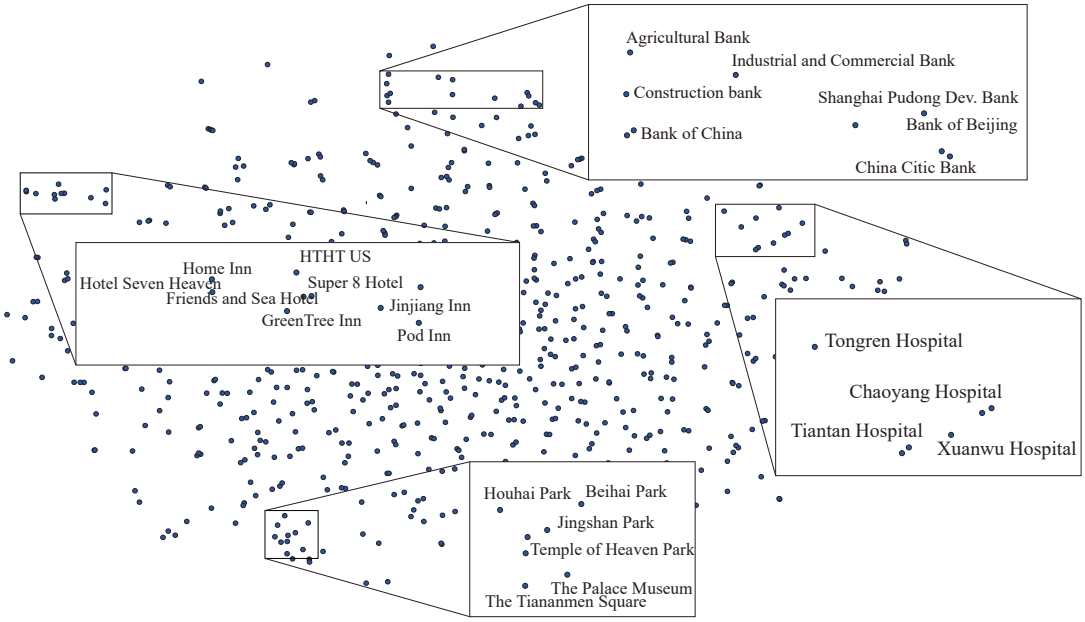


Fig. 8: The embeddings of the learned vectors of map queries in the two-dimensional space.

[16] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 553–562.

[17] S. Ghosh, O. Vinyals, B. Strope, S. Roy, T. Dean, and L. Heck, "Contextual lstm (clstm) models for large scale nlp tasks," *arXiv preprint arXiv:1602.06291*, 2016.

[18] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[19] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. of CVPR*, 2015, pp. 3128–3137.

[20] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," 2016.

[21] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng, "Learning user reformulation behavior for query auto-completion," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 445–454.

[22] G. E. Hinton, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, vol. 1. Amherst, MA, 1986, p. 12.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.

[24] A. Paccanaro and G. E. Hinton, "Learning distributed representations of concepts using linear relational embedding," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 13, no. 2, pp. 232–244, 2001.

[25] W. Xu and A. I. Rudnicky, "Can artificial neural networks learn language models?" Computer Science Department, 2000.

[26] K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*. Association for Computational Linguistics, 2000, pp. 63–70.

[27] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

[28] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 873–882.

[29] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model." in *SLT*, 2012, pp. 234–239.

[30] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
[32] B. Mitra, "Exploring session context using distributed representations of queries and reformulations," in *Proc. of the 38th ACM SIGIR on Research and Development in Information Retrieval*. ACM, 2015, pp. 3–12.
[33] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proc. of the 19th WWW*. ACM, 2010, pp. 811–820.
[34] N. Natarajan, D. Shin, and I. S. Dhillon, "Which app will you use next?: collaborative filtering with interactional context," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 201–208.
[35] J. W. Jun Chen, Chaokun Wang, "A personalized interest-forgetting markov model for recommendations," in *29th AAAI*, 2015.
[36] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, "Context-aware query suggestion by mining click-through and session data," in *Proc. of the 14th ACM SIGKDD*. ACM, 2008, pp. 875–883.
[37] R. Baraglia, F. M. Nardini, C. Castillo, R. Perego, D. Donato, and F. Silvestri, "The effects of time on query flow graph-based models for query suggestion," in *Adaptivity, Personalization and Fusion of Heterogeneous Information*, 2010, pp. 182–189.
[38] J. Wang, J. Z. Huang, J. Guo, and Y. Lan, "Recommending high-utility search engine queries via a query-recommending model," *Neurocomputing*, vol. 167, pp. 195–208, 2015.
[39] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
[40] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.
[41] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, D. Blei and F. Bach, Eds. JMLR Workshop and Conference Proceedings, 2015, pp. 2048–2057. [Online]. Available: http://jmlr.org/proceedings/papers/v37/xuc15.pdf
[42] J. Huang and E. N. Efthimiadis, "Analyzing and evaluating query reformulation strategies in web search logs," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 77–86.
[43] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." Citeseer, 2011.
[44] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.
[45] D. R. Radev, H. Qi, H. Wu, and W. Fan, "Evaluating web-based question answering systems," *Ann Arbor*, vol. 1001, p. 48109, 2002.
[46] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

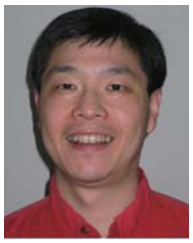
Jun Song received B.E. degree from Tianjin University, Tianjin, China, in 2013. He is currently a Ph.D. candidate in computer science with the Digital Media Computing and Design Lab of Zhejiang University. His research interests include machine learning, cross-media information retrieval and understanding.

Jun Xiao received the Ph.D. degree in computer science and technology from the College of Computer Science, Zhejiang University (ZJU), Hangzhou, China, in 2007. He is currently an Associate Professor with the Microsoft Visual Perception Laboratory, College of Computer Science, ZJU. His current research interests include computer animation, digital entertainment, and multimedia retrieval.

Fei Wu received the B.Sc. degree from Lanzhou University, Lanzhou, China, in 1996, the M.Sc. degree from the University of Macau, Macau, China, in 1999, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2002, all in computer science. He is currently a full Professor with the College of Computer Science and Technology, Zhejiang University. His current research interests include multimedia retrieval, sparse representation, and machine learning.

Haishan Wu is currently a data scientist and team lead of spatial-temporal big data group in Big Data Lab of Baidu Research Institute, focusing on mining spatial-temporal mobile big data of Baidu (mobile location data, trajectory data, location query data from Baidu Maps) for both scientific research and business applications. He received B.E. degree from China University of Mining and Technology, in 2006, and the Ph.D. degree from Fudan University, Shanghai, China.

Tong Zhang received Ph.D. degree from Stanford Computer Science. After graduation, He worked at IBM T.J. Watson Research Center in Yorktown Heights, New York, and Yahoo! Research in New York City. He is currently with the statistics department, Rutgers University and vice president of Baidu Research Institute.



Zhongfei (Mark) Zhang received the BS (cum laude) degree in electronics engineering, the MS degree in information science, both from Zhejiang University, and the Ph.D. degree in computer science from the University of Massachusetts at Amherst. He is currently a full professor of computer science the State University of New York (SUNY) at Binghamton. He directs the Multimedia Research Laboratory at Binghamton.

Wenwu Zhu (M97 SM01 F10) is currently a Professor and Deputy Head of Computer Science Department of Tsinghua University. Wenwu Zhu is an IEEE Fellow, SPIE Fellow, and ACM Distinguished Scientist. He has published over 200 referred papers in the areas of multimedia computing, communications and networking. He is inventor or co-inventor of over 50 patents. His current research interests are in the area of multimedia big data computing, social multimedia computing, multimedia cloud computing, and multimedia communications and networking. He received the Ph.D. degree from New York University Polytechnic School of Engineering in 1996 in Electrical and Computer Engineering.