PSTIE: Time Information Enhanced Personalized Search

Zhengyi Ma[2], Zhicheng Dou[1], Guanyue Bian[2], and Ji-Rong Wen[3,4]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Key Laboratory of Data Engineering and Knowledge Engineering, MOE
zymaa@ruc.edu.cn,dou@ruc.edu.cn,yueliang0104@ruc.edu.cn,jirong.wen@gmail.com

ABSTRACT

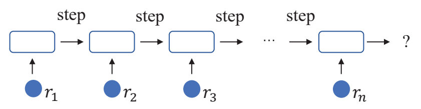
Personalized search aims to improve the quality of search results by re-ranking the candidate document list based on the historical behavior of each user. Existing approaches focus on modeling the order information of the user's historical behaviors by sequential methods such as Recurrent Neural Network (RNN). However, these methods usually ignore the fine-grained time information associated with user actions. In fact, the time intervals between queries can help to capture the evolution of query intent and document interest of users. Besides, the time intervals between past actions and current query can reflect the long-term re-finding interest more accurately than discrete steps in RNN and its variants. In this paper, we propose **PSTIE**, a fine-grained **T**ime **I**nformation **E**nhanced model for constructing more accurate user interest representations for **P**ersonalized **S**earch. To capture the short-term interest of users, we design time-aware LSTM architectures for modeling the subtle interest evolution of users in continuous time. We further leverage time in calculating the re-finding possibility of each historical query and document to capture the long-term interest of users. We propose two methods to utilize the time-enhanced user interest in personalized ranking. Extensive experiments on two real-world datasets show that our approach can effectively improve the quality of search results over state-of-the-art models.

KEYWORDS

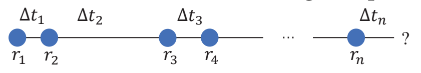
Personalized Search, Time Information, User Interest, Re-finding

1 INTRODUCTION

Traditional search engines give the same results to different users when they issue the same query. However, this strategy cannot distinguish different information needs of different users, such as the query "Apple" (Apple fruit or Apple company?). Personalized search aims to improve the quality of search results by re-ranking the candidate document list based on the historical behaviors of users. Traditional studies of personalized search manually extract features from search logs of each user [4–6, 8, 11, 15, 22, 29, 33]. However, these feature-based methods cannot cover all aspects of



(a) RNN architectures for modeling user profiles



(b) Behavior of Users in Personalized search

Figure 1: r_i is the representation of user behavior at the i-th query q_i, and Δt_i refers to the time interval between q_i and q_{i+1}. As shown in Figure 1(a), existing personalized methods based on RNN ignored time information Δt_i.

the data distribution. Deep learning can automatically learn representations of queries, documents, and users, then capture user interest dynamics from large scale user behavior data automatically [14, 20, 21, 30, 37, 39]. Inspired by successful applications of RNN in natural language processing tasks, most existing neural personalized search methods build distributed representations of user interest with RNN or its variants as shown in Figure 1(a). They then re-rank the document list based on the similarity of computed interest representation and document representation. These RNN-based sequential methods [14, 21, 37] have been proved effective and significantly improved search quality.

However, none of the above methods has considered explicit time information, such as time intervals between historical queries, as shown in Figure 1(b). They mainly considered the relative order of user behaviors. In fact, time intervals between past queries can help model user interests more accurately. For example, if the user issues another query 30 days after a previous query, these two queries may have little correlations between each other. Thus, the former query and the corresponding documents tend to have relatively lower influence when the user issues the latter query. Conversely, two queries within a short time, like 5 minutes, tend to be more relevant. **Thus, user's interest evolution within a short time is time-sensitive.** Traditional RNN architectures are good at modeling the order information of sequential data of users [14]. However, they cannot distinguish the above phenomenon because they ignore time interval information between actions. Therefore, the interest representations constructed by traditional RNN architectures can be inaccurate. Besides, several personalized methods group historical user behaviors into sessions based on time intervals between queries (for example, splitting sessions based on 30-minutes inactivity [5, 11, 14, 21]). These strategies can lead to propagation

CIKM '20, October 19–23, 2020, Virtual Event, Ireland
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411877

errors in the user interest modeling module due to some incorrectly identified sessions. The above problem can be alleviated if we can directly exploit the time information in sequential modeling without identifying sessions.

In the meanwhile, users tend to have long-term re-finding behavior. They often seek some information they have encountered before after a relatively long time. Previous studies tried to help users re-find documents they are interested in, by measuring the lexical or semantic similarities of documents to historical user behaviors [31, 32]. However, they cannot track how the tendency of re-finding behavior fluctuates with time explicitly. As a result, they fail to capture various temporal dynamics of the user's interests. For example, some users tend to visit the official website of the "CIKM" conference once a year due to its yearly-held feature. To find the website, they may issue "CIKM" or its related queries like "CIKM 2020" once a year. In fact, users tend to seek some information again near the end of its lifetime. **Thus, re-finding behavior is time-sensitive.** We believe that considering time-sensitive re-finding behavior is useful for learning more accurate interests of users. However, most existing studies have ignored this.

In this paper, we present PSTIE: a **T**ime **I**nformation **E**nhanced model for **P**ersonalized **S**earch, which leverages the fine-grained time information between search behaviors of users to improve ranking quality. We exploit time information to construct two kinds of time-sensitive representations: time-sensitive query intent and time-sensitive document interest. The query intent representation can help personalized models understand user's current query more accurately, while document interest can reflect the user's preference for documents. We classify both representations into short- and long-term representations based on the two kinds of time-sensitive behaviors of users that we mentioned above. Short-term query intent refers that users tend to have similar intent within a short time and issue a related query since the last query. Long-term query intent means that the user's intent follows a lifetime distribution that drifts with time, and users tend to issue a similar query near the lifetime of the intent. Short-term document interest refers that users tend to click similar documents within a short time, while long-term one means that the re-finding for similar documents follows a lifetime distribution drifting with time.

To model the time-sensitive **short-term interest**, we design two time-aware LSTM architectures to separately model the evolution of query intent and document interest in continuous time. To model the time-sensitive **long-term interest**, we calculate the probability of long-term re-finding behavior by a query-specific Gaussian mixture distribution that drifts with time. Besides, we introduce two matching methods to fuse the time-sensitive interest of users into the ranking module. The first one is using the cosine similarity of interest representation and document representation as to the time-sensitive personalized score. The second one is using the query intent representations and document interest representations to initialize the hidden states of the LSTM network in our interactive matching model. Experimental results on two real-world datasets show that our model can effectively improve search quality. This indicates that modeling fine-grained time information in user profiling is beneficial to personalized search.

Our main contributions are three-fold: (1) We leverage fine-grained time information within user's historical behaviors to improve personalized ranking quality. To the best of our knowledge, this is the first study where fine-grained time information is considered for personalized search. (2) We track two kinds of time-sensitive evolution of users, including query intent evolution and document interest evolution. We consider both short-term local correlations and long-term re-finding influences between user's search history. (3) We use two matching methods, a representation-based matching and an interaction-based matching, to fuse the time-sensitive interest representations into personalized ranking.

The rest of paper is organized as follows. We introduce related works in Section 2. We then introduce our time-enhanced personalized model in Section 3. We describe experimental settings and results in Section 4, then conclude the paper in Section 5.

2 RELATED WORK

2.1 Personalized Search

Personalized search aims to return personalized document list based on user interests [7]. Traditional personalized methods focus on modeling the influence of click behavior and topic features [4, 8, 11, 15, 22, 29, 33]. To combine these features, some studies proposed to utilize learning to rank algorithm for training a better ranking model [5, 6]. However, these methods suffered from heavy burden for its manually extracted features, and they cannot learn abstract sequential and temporal factors automatically.

Deep learning can learn distributed representations of queries, documents, and even users automatically [16, 18, 24, 25, 28, 37, 39]. Because of this, it has been utilized into personalized search by some previous studies and achieved adorable results [14, 20, 21, 30]. Song et al. [30] introduced a general RankNet framework for personalized search. Li et al. [20] improved the personalized performance by applying semantic features from in-session context into models. Ge et al. [14] proposed a sequential method based on hierarchical RNN, considering both long- and short-term interest and applying query-aware attention to interact between the current query and historical sessions. Lu et al. [21] trained a generative adversarial network with GRU to make the model more robust with noisy click data. These RNN based methods are good at capturing sequential information, especially order information, but they cannot model the time intervals between queries explicitly. We can model more accurate interest representations of users by modeling the fine-grained time information into our neural sequential model.

2.2 Sequential Models with Time Information

While RNN and its variants achieved great success in capturing sequential patterns automatically, it only considers the order information and ignores the time intervals between events. Several works have proved that time intervals are important to capture the correlations between actions [3, 17, 40]. One naive and general method to deal with continuous-time features is to bucketize the time according to the time distribution [9]. However, it brings the unwanted discretization error, which formally refers to the Modifiable Areal Unit Problem, as revealed by [12]. Several studies [38, 40] designed gates in LSTM to model the event correlations based on time intervals or distances. These gated models are easy to employ

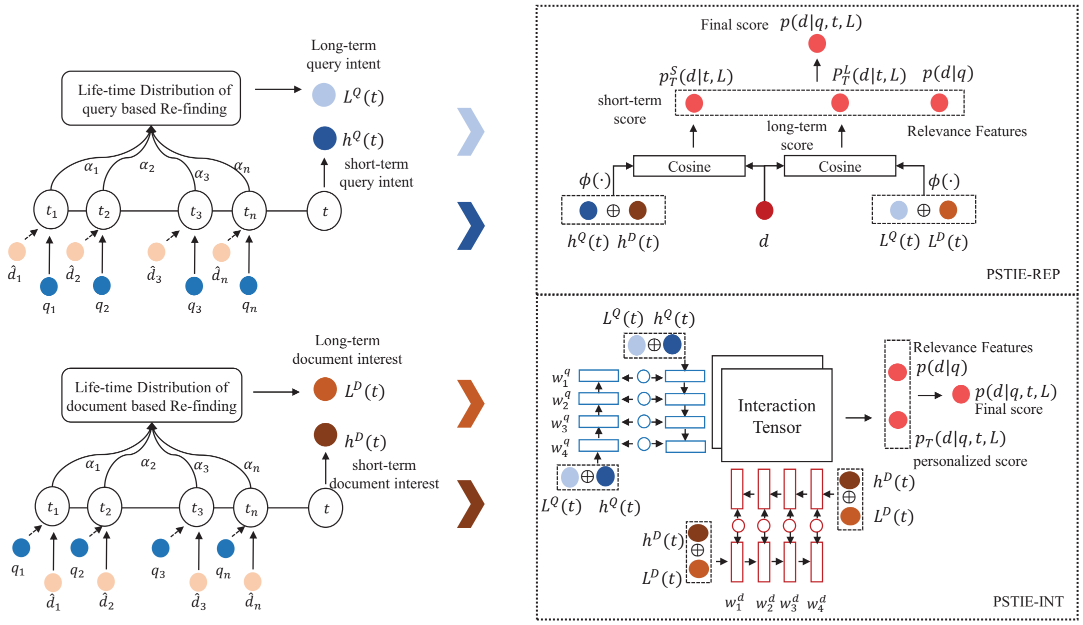


Figure 2: The architecture of PSTIE. Given search history, we design two time-aware LSTM architectures to model interest evolution drifting with time, and calculate the short-term query intent and document interest representations at the current time. After modeling the probability of two kinds of re-finding behavior by a lifetime distribution, long-term query intent and document interest are collected. Two methods are designed for combining interest representations into re-ranking.

and inherit from the powerful ability of RNN to model sequential patterns. However, it cannot explicitly model complex temporal patterns like lifetime re-finding distributions of users, and usually show poor performance when the sequence is long.

Another way to exploit time information is to model the discrete events in continuous time. These approaches can directly absorb the raw timestamp of each discrete event, avoiding loss of time information. Bai et al. [3] designed a neural demand-aware Hawkes process framework to capture the long- and short-term demands of users in continuous time based on Mei and Eisner [23]. However, it cannot model the lifetime demands of users explicitly and interpretably. Wang et al. [35] factorized the repeat consumption influence as a short-time and lifetime effect, and introduced the Hawkes process into collaborative filtering. However, the personalized search problem is relatively more difficult than recommendation systems since we have to consider the complex interaction between queries, documents, click information, and others.

3 METHODOLOGY

3.1 Problem Formulation

Personalized search aims to construct interest representations of users to re-rank candidate documents. Assume we have a user set $U = \{u_1, u_2, ...u_M\}$. For each user u, we have its query log $L_u = \{(\boldsymbol{q}_1, \boldsymbol{D}_1, t_1), (\boldsymbol{q}_2, \boldsymbol{D}_2, t_2), ...(\boldsymbol{q}_{n_u}, \boldsymbol{D}_{n_u}, t_{n_u})\}$, where $(\boldsymbol{q}_i, \boldsymbol{D}_i, t_i)$ refers that user u issued the i-th query $\boldsymbol{q}_i$ at time t_i and get the search results $\boldsymbol{D}_i$. Moreover, $\boldsymbol{D}_i$ can be defined as $\boldsymbol{D}_i = \{\boldsymbol{d}_{i1}, ...\boldsymbol{d}_{im}\}$, where $\boldsymbol{d}_{ij}$ refers to the j-th document in the candidate document list of $\boldsymbol{q}_i$. Each candidate document contains a clicked label to record whether it is clicked by u. We also calculate the average of the clicked documents under query $\boldsymbol{q}_i$ as $\hat{\boldsymbol{d}}_i$. Until now, we get the search history of user u. For simplicity, we omit the subscript of u and use $L = \{(\boldsymbol{q}_1, \boldsymbol{D}_1, t_1), ...(\boldsymbol{q}_n, \boldsymbol{D}_n, t_n)\}$ to denote the search history of a user, where n represents the number of queries in uses's history. We define the output of our model as $p(\boldsymbol{d}|\boldsymbol{q}, t, L)$, where $\boldsymbol{q}$ represents the current query that user issues at current time t and $\boldsymbol{d}$ represents a candidate document that has been retrieved by the search engine based by query $\boldsymbol{q}$. The output $p(\boldsymbol{d}|\boldsymbol{q}, t, L)$ represents the final score of document $\boldsymbol{d}$ for the user. We calculate $p(\boldsymbol{d}|\boldsymbol{q}, t, L)$ for every document $\boldsymbol{d}$ in the candidate document list and re-rank the list by $p(\boldsymbol{d}|\boldsymbol{q}, t, L)$ in a descending order. Specifically, the output of our model is:

$$p\left(\boldsymbol{d}|\boldsymbol{q}, t, L\right) = \phi\left(p_T\left(\boldsymbol{d}|\boldsymbol{q}, t, L\right), p\left(\boldsymbol{d}|\boldsymbol{q}\right)\right), \quad (1)$$

where $p_T\left(\boldsymbol{d}|\boldsymbol{q}, t, L\right)$ represents time-sensitive personalized score of document $\boldsymbol{d}$ at current time t, and $p(\boldsymbol{d}|\boldsymbol{q})$ refers to the ad-hoc relevance between the document $\boldsymbol{d}$ and query $\boldsymbol{q}$. $\phi(\cdot)$ is a Multi-layer Perception (MLP) with tanh$(\cdot)$ as an activation function. We use MLP to learn the weights of the two parts and combine them to get the final score of a document.

3.2 PSTIE: the Overall Framework

The architecture of our PSTIE framework is shown in Figure 2. Since RNN cannot model the time intervals between historical behaviors explicitly, we design two time-aware LSTM architectures for modeling the sequence of queries and clicked documents in continuous time. We get the query intent and document interest representations of the user at the current time via time-aware LSTM sequences.

As these representations focus on the local interest of users within a short time, we call them the short-term interest of users. As we have illustrated in Section 1, users tend to have re-finding behavior near the end of the lifetime of information. We further calculate the contribution of historical queries and documents to the long-term re-finding behavior based on the query-specific Gaussian mixture distribution that drifts with time. We calculate the long-term query intent and document interest of users by summing up the historical information based on the contribution we calculated. For the final personalized re-ranking, we introduce two methods to combine the interest of users with the current query and document for unified document scoring. We calculate the final score of the candidate document based on the personalized score and ad-hoc relevance score as Equation (1).

Before introducing the detailed model, we firstly represent the historical queries and documents based on word embedding. We train the word embedding on the query log with Glove [27]. Then we calculate the weighted average of word embedding by TF-IDF weights as the embedding of queries and documents.

3.3 Short-term Interest Modeling with Time

We wish to leverage time intervals between the search history of users to improve ranking quality. Inspired by [23], we design the framework of document-driven time-aware LSTM for modeling query sequences, and query-driven time-aware LSTM for clicked document sequences. Differently from [23], the application scenario of our model is web search, which has more complex correlations between queries, documents, and users than recommendation systems. Thus, we design two architectures considering the special characteristic of search behavior to model the intent and interest evolution of users that fluctuates with time. We use time-aware LSTM to calculate the short-term interest of users at the current time. We believe that the query intent representation we calculate in continuous time can reflect the query that the user is going to issue. In the meanwhile, the document interest calculated by query-driven time-aware LSTM can reflect the preference of users for documents.

3.3.1 Document-driven Time-aware LSTM for Query Intent. When a user issues a query in a search engine, it can be predicted that he is describing his information need through this query and he may probably hold similar intent at several next queries. We can see from the above phenomenon that the query intent of users shows self-exciting characteristic [3, 23]. In the meanwhile, the intent of users will decay with time because users will have different information need later. The characteristic of query intent drifting with time inspires us to design an architecture that can accurately model the intent evolution. Based on the above phenomenon, we design document-driven time-aware LSTM, which leverages time intervals between query sequence $\{\boldsymbol{q}_1, \boldsymbol{q}_2, ...\boldsymbol{q}_n\}$ to model the intent evolution of users. The hidden state vector $\boldsymbol{h}(t)$ of time-aware LSTM is calculated in continuous time as follows:

$$\boldsymbol{h}(t) = \boldsymbol{o}_k \odot [2\sigma(2\boldsymbol{c}(t)) - 1], \tag{2}$$

where $\boldsymbol{o}_k$ refers to the output gate of time-aware LSTM and $t \in [t_k, t_{k+1}]$. At each time step t_i, the time-aware LSTM will treat the i-th query $\boldsymbol{q}_i$ as its input, and update the $\boldsymbol{c}(t)$ to a new initial value $\boldsymbol{c}_{i+1}$ as follows:

$$\boldsymbol{c}_{i+1} \leftarrow \boldsymbol{f}_{i+1} \odot \boldsymbol{c}(t_i) + \boldsymbol{i}_{i+1} \odot \boldsymbol{z}_{i+1}, \tag{3}$$

where $\boldsymbol{f}_{i+1}, \boldsymbol{i}_{i+1}, \boldsymbol{z}_{i+1}$ refers to the forget gate, input gate and candidate memory in time-aware LSTM, and they are calculated based on the input $\boldsymbol{q}_i$ and hidden state $\boldsymbol{h}(t_i)$ at each time step following [23]. The jump of $\boldsymbol{c}(t)$ to a specific value at each time step models the self-exciting characteristic of the user's search behaviour. As t increases between t_i and t_{i+1}, user's need of information related to the query $\boldsymbol{q}_i$ will decay. Specifically, the cell memory $\boldsymbol{c}(t)$ will decay exponentially from $\boldsymbol{c}_{i+1}$ towards target $\overline{\boldsymbol{c}}_{i+1}$ as follows:

$$\overline{\boldsymbol{c}}_{i+1} \leftarrow \overline{\boldsymbol{f}}_{i+1} \odot \overline{\boldsymbol{c}}_i + \overline{\boldsymbol{i}}_{i+1} \odot \boldsymbol{z}_{i+1} + \overline{\boldsymbol{d}}_{i+1} \odot \hat{\boldsymbol{d}}_i, \tag{4}$$

$$\boldsymbol{c}(t) = \overline{\boldsymbol{c}}_{k+1} + (\boldsymbol{c}_{k+1} - \overline{\boldsymbol{c}}_{k+1}) \exp(-\delta_{k+1}(t - t_k)), \tag{5}$$

where the target cell $\overline{\boldsymbol{c}}_{i+1}$ is not only controlled by the previous cell state $\overline{\boldsymbol{c}}_i$, but is also driven by the clicked documents $\hat{\boldsymbol{d}}_i$ under the query $\boldsymbol{q}_i$. This is based on the fact that the clicked documents can satisfy user's information need after the issued query $\boldsymbol{q}_i$. Specifically, we tailor $\overline{\boldsymbol{c}}_{i+1}$ in Equation (4) with a document gate $\overline{\boldsymbol{d}}_{i+1}$ and the average of clicked documents $\hat{\boldsymbol{d}}_i$ at t_i. The document gate $\overline{\boldsymbol{d}}_{i+1}$ is calculated as follows:

$$\overline{\boldsymbol{d}}_{i+1} \leftarrow \sigma(\boldsymbol{W}_d^1 \boldsymbol{q}_i + \boldsymbol{W}_d^2 \hat{\boldsymbol{d}}_i + \boldsymbol{U}_d \boldsymbol{h}(t_i) + \boldsymbol{d}_d), \tag{6}$$

where $\boldsymbol{W}_d^1$, $\boldsymbol{W}_d^2$, $\boldsymbol{U}_d$ and $\boldsymbol{d}_d$ are parameter matrices and vector to be learned for calculating the document gate $\overline{\boldsymbol{d}}_{i+1}$.

Now we have modeled the query sequence with document-driven time-aware LSTM as Equation (2)-(6). We calculate the hidden states $\boldsymbol{H}_q = \{\boldsymbol{h}^q(t_1), \boldsymbol{h}^q(t_2)...\boldsymbol{h}^q(t_n)\}$ at each time step as the context-aware representation of historical query intent of user. We also calculate the hidden state $\boldsymbol{h}^q(t)$ at current query time t as the short-term query intent representation. $\boldsymbol{h}^q(t)$ contains the local information need of user at current time.

3.3.2 Query-driven Time-aware LSTM for Document Interest. Besides query intent, clicked documents are also crucial for building more accurate user interest representations. We should build document interest representations that reflect the user's preference for documents. To model the document interest evolution that drifts with time, we design query-driven time-aware LSTM for modeling the clicked documents sequence $\{\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, ...\hat{\boldsymbol{d}}_n\}$ in continuous time. The document interest also has characteristics of self-exciting and decaying. Specifically, the user's preference for documents will jump discontinuously when they have read some certain documents, and the interest for those documents will decay with time. Besides, the document interest should be influenced by the corresponding queries, because the query straightly reflects the information need of users. As a result, the document interest should jump to a certain initial value based on the issued query at each time step. As time decays, the document interest will decay while users are reading the documents. Specifically, the initial value $\boldsymbol{c}_{i+1}$ that document interest will jump to depends on the issued query as follows:

$$\boldsymbol{c}_{i+1} \leftarrow \boldsymbol{f}_{i+1} \odot \boldsymbol{c}(t_i) + \boldsymbol{i}_{i+1} \odot \boldsymbol{z}_{i+1} + \overline{\boldsymbol{q}}_{i+1} \odot \boldsymbol{q}_i, \tag{7}$$

$$\overline{\boldsymbol{c}}_{i+1} \leftarrow \overline{\boldsymbol{f}}_{i+1} \odot \overline{\boldsymbol{c}}_i + \overline{\boldsymbol{i}}_{i+1} \odot \boldsymbol{z}_{i+1}, \tag{8}$$

where the calculation of hidden state $\boldsymbol{h}(t)$ and cell memory $\boldsymbol{c}(t)$ is the same as Equation (2) and (5). $\overline{\boldsymbol{q}}_{i+1}$ represents the query gate that controls the influence of the issued query, which is defined as:

$$\overline{\boldsymbol{q}}_{i+1} \leftarrow \sigma(\boldsymbol{W}_q^1 \boldsymbol{q}_i + \boldsymbol{W}_q^2 \hat{\boldsymbol{d}}_i + \boldsymbol{U}_q \boldsymbol{h}(t_i) + \boldsymbol{q}_q), \quad (9)$$

where $\boldsymbol{W}_q^1$, $\boldsymbol{W}_q^2$, $\boldsymbol{U}_q$ and $\boldsymbol{q}_q$ are parameter matrices and vector to be learned for calculating the query gate $\overline{\boldsymbol{q}}_{i+1}$. We organize the clicked documents based on queries for two reasons: (1) The clicked documents under each query tend to be semantically similar. Thus, they can better reflect the overall document preference of users and alleviate some misclick noise. (2) The number of documents in one real-world search engine is too large. As a result, it will lead to poor performance because of the sparsity.

Using query-driven time-aware LSTM, we calculate the hidden states of document sequence $\boldsymbol{H}_d = \{\boldsymbol{h}^d(t_1), \boldsymbol{h}^d(t_2), ...\boldsymbol{h}^d(t_n)\}$ as the context-aware representations of document interest. We further calculate $\boldsymbol{h}^d(t)$ as the short-term document interest vector, which represents the user's preference for documents within a short time.

3.4 Long-term Re-finding Interest Modeling

The short-term interest representations straightly represent query intent and document interest of users within a short time. However, time-aware LSTM can hardly capture the influence of queries or documents if they were issued or clicked a long time ago. It is a common shortcoming of RNN-based sequential methods. As we illustrated in Section 1, users usually have time-sensitive re-finding behavior following lifetime distribution. Specifically, users tend to show similar query intent or document interest near the end of the lifetime of information. The probability of re-finding will rise with time after the last query or reading documents, and reach its top value at the lifetime of information. After the lifetime, the re-finding probability of the user will decay with time. Inspired by [35], we choose a Gaussian mixture distribution to model the above long-term re-finding interest evolution of users.

It is natural to use Gaussian mixture distribution to model the lifetime evolution of re-finding behavior drifting with time. Besides, it has good interpretability to explain the lifetime and influence of information. We consider two kinds of re-finding behavior in our model: query-based re-finding and document-based re-finding. The former means that users may try to follow a specific topic with similar queries. The latter focuses on the user's interest in a specific document. As we have introduced in the previous Section 3.3.2, we use the query-specific lifetime distribution to model the two kinds of re-finding interest. Specifically, each query $\boldsymbol{q}_i$ has a specific parameter set $\{\mu_i, \sigma_i\}$ for its lifetime distribution. We calculate the probability of re-finding behavior α_i based on the query-specific Gaussian mixture distribution as follows:

$$\alpha_i = N(\delta t_i | \mu_i, \sigma_i), \quad (10)$$

where μ^i can represent the lifetime of query intent and document interest at the i-th query, and σ_i can reflect the influence of the re-finding. α_i can reflect the re-finding probability of the i-th query intent or document interest, and it varies with the time interval $\delta t_i = t - t_i$, where t is the current time and t_i is the issued time of query $\boldsymbol{q}_i$. We sum up the history intent representations weighted by the softmax of the re-finding probability, and calculate the long-term query intent vector $\boldsymbol{L}^q(t)$. Specifically,

$$\boldsymbol{L}^q(t) = \sum_{i=1}^{n} \frac{\exp(\alpha_i)}{\sum_{j=1}^{n} \exp(\alpha_j)} \boldsymbol{h}^q(t_i), \quad (11)$$

where n is the number of queries in history, and $\boldsymbol{h}^q$ is the historical query intent representations we calculate in the previous section. We also calculate the weighted sum of the historical document interest representations to get the long-term document interest vector $L^d(t)$ as follows:

$$\boldsymbol{L}^d(t) = \sum_{i=1}^{n} \frac{\exp(\alpha_i)}{\sum_{j=1}^{n} \exp(\alpha_j)} \boldsymbol{h}^d(t_i). \quad (12)$$

Now, we have got the long-term query intent and document interest representations. The long-term query intent will attend more on the historical information need that users tend to pose at the current time. The long-term document interest focus on some documents that users want to re-seek.

3.5 Time-Sensitive Personalized Ranking

Now we have calculated out the short-term query intent vector $\boldsymbol{h}^q(t)$, the short-term document interest vector $\boldsymbol{h}^d(t)$, the long-term query intent vector $\boldsymbol{L}^q(t)$ and the long-term document interest vector $\boldsymbol{L}^d(t)$ at time t with the current issued query $\boldsymbol{q}$. We will utilize these interest representations to calculate the time-sensitive personalized score $p_T(\boldsymbol{d}|\boldsymbol{q}, t, L)$ for each document $\boldsymbol{d}$, and finally calculate the final score $p(\boldsymbol{d}|\boldsymbol{q}, t, L)$. We will introduce two different methods to utilize the personalized interest representations into the personalized ranking, which are shown in Figure (2).

3.5.1 *Representation-based Similarity.*

The first method is based on representation. We will concatenate the short-term query intent vector $\boldsymbol{h}^q(t)$ with the short-term document interest vector $\boldsymbol{h}^d(t)$ as the final short-term user interest. The long-term user interest is calculated by concatenating the long-term query intent $\boldsymbol{L}^q(t)$ and long-term document interest $\boldsymbol{L}^d(t)$. We calculate the personalized score of document $\boldsymbol{d}$ by measuring the similarity between the document vector and two final interest vectors.

$$p_T^S(\boldsymbol{d}|\boldsymbol{q}, t, L) = \text{sim}\left(\phi\left([\boldsymbol{h}^q(t); \boldsymbol{h}^d(t)]\right), \boldsymbol{d}\right), \quad (13)$$

$$p_T^L(\boldsymbol{d}|\boldsymbol{q}, t, L) = \text{sim}\left(\phi([\boldsymbol{L}^q(t); \boldsymbol{L}^d(t)]), \boldsymbol{d}\right), \quad (14)$$

where [;] means a concatenation. We apply an MLP layer to the concatenated interest to maintain its same semantic space as the document. $p_T^S(\boldsymbol{d}|\boldsymbol{q}, t, L)$ represents the short-term personalized score and $p_T^L(\boldsymbol{d}|\boldsymbol{q}, t, L)$ is the long-term personalized score. Many functions for measuring similarity between two distributed vectors can be used for the similarity function in the above equation. In this work, we use Cosine as our similarity function, which is commonly used for modeling interactions. It is viewed as the angle of two vectors:

$$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^T \boldsymbol{v}}{||\boldsymbol{u}|| \cdot ||\boldsymbol{v}||}. \quad (15)$$

We use MLP to learn the weights of short- and long-term personalized score, then calculate the final score $p_T(\boldsymbol{d}|\boldsymbol{q}, t, L)$ by:

$$p_T(\boldsymbol{d}|\boldsymbol{q}, t, L) = \phi(p_T^S(\boldsymbol{d}|t, L), p_T^L(\boldsymbol{d}|t, L)). \quad (16)$$

The representation-based re-ranking approach is commonly used in previous works [14, 21]. However, it cannot naturally incorporate the current query $\boldsymbol{q}$ into the scoring module. In fact, the word-level information in query and document can bring fine-grained matching features and help to calculate more accurate similarity. However, the representation-based method cannot leverage it since it has compressed query and document into distributed vectors.

3.5.2 Interaction-based Matching. Interactive matching applies pooling strategy across the similarity matrices, then calculates the matching score between query and document. Since it calculates word-level similarity score, it can detect some fine-grained matching signals between words in queries and documents. Inspired by [34], we build our model based on MV-LSTM. Compared to another well-known matching method K-NRM [36], MV-LSTM can calculate multiple positional sentence representations for query and document dynamically. Besides, we can fuse the interest representations into interactive matching more naturally. As shown in Figure (2), we concatenate the short-term query intent $\boldsymbol{h}^q(t)$ with long-term query intent $\boldsymbol{L}^q(t)$ as the final query intent vector, and use the final query intent representation to initialize the hidden state of Bidirectional LSTM of queries in MV-LSTM. We believe the query intent can help the Bidirectional LSTM to calculate more accurate sentence representations at each position. Similarly, we use the concatenated vector of short-term document interest $\boldsymbol{h}^d(t)$ and long-term document interest $\boldsymbol{L}^d(t)$ to initialize the hidden state of document Bidirectional LSTM.

Since we use interest representations to initialize the calculation of the positional encoding of queries and documents, our model can help to solve the ambiguity problem in personalized search. For example, when a user issues a query "apple", it can be interpreted as both the apple company or the fruit apple. If one user has issued many queries about apple company or its related products, we can calculate the query intent vector that preferred the semantic meaning "apple company". Then, the query intent vector can help the bidirectional LSTM to calculate the query representation that is semantically more related to the preferred meanings of users. In the meanwhile, the document interest vector can help to compute more accurate document representations that focus more on the aspects that the user may be interested in documents.

3.6 Model Training

Until now, we have defined the personalized score of one candidate document $\boldsymbol{d}$ as $p_T(\boldsymbol{d}|\boldsymbol{q}, t, L)$. For ad-hoc relevance $p(\boldsymbol{d}|\boldsymbol{q})$, we follow the idea of [5] and extract several traditional topic features and click features for every document. We feed the feature vector of document $\boldsymbol{d}$ to a multi-layer perception(MLP) with tanh(·) activate function to calculate the base score of one document:

$$p(\boldsymbol{d}|\boldsymbol{q}) = \phi(\boldsymbol{W}^F f_{q,d} + b^F). \quad (17)$$

Now we can calculate the final score $p(\boldsymbol{d}|\boldsymbol{q}, t, L)$ for every document $\boldsymbol{d}$ in our candidate list based on Equation (1). We get the re-ranked list of document based on the score as the output of our model. As for training, we apply the learning-to-rank framework with a pair-wise strategy in our model. We select a satisfying clicked document as a positive sample and a skipped document as a negative sample. We wish to maximize the distance of the positive score and the negative score. Thus, we use a weighted cross-entropy to measure the loss between the true distance $dis_{i,j}^T$ and the predicted distance $dis_{i,j}^P$, i.e.,

$$loss = -\lambda_{i,j}(dis_{i,j}^T \log(dis_{i,j}^P) + (1 - dis_{i,j}^T \log(1 - dis_{i,j}^P)), \quad (18)$$

where we use the change of swapping the positions of the two documents as $\lambda_{i,j}$ as the weights motivated by LambdaRank [6]. We calculate the score distance between two documents as follows:

$$dis_{i,j} = \frac{1}{1 + \exp(p(\boldsymbol{d}_j|\boldsymbol{q}, t, L) - p(\boldsymbol{d}_i|\boldsymbol{q}, t, L))}. \quad (19)$$

In summary, we introduce a framework to leverage the fine-grained time information between search behaviors of users to build more accurate user interest representations. We design two time-aware LSTM based on the characteristic of search behavior for building short-term user interest in continuous time, including query intent and document interest. We further integrate the lifetime characteristic of re-finding behavior into our model and build long-term interest of users. We propose two methods to combine the interest of users for document re-ranking, including representation-based similarity and interaction-based matching. We rank the document list based on the final score we calculated for each document.

Table 1: Statistics of the Datasets

AOL		Commercial	
Item	Statistic	Item	Statistic
days	92	days	58
users	118,067	users	7,648
queries	3,461,636	queries	694,837
distinct queries	1,555,829	distinct queries	278,661
SAT-clicks	4,701,531	SAT-clicks	443,428
Co-queries	8,184,227	Co-query	4,109,396
Re-queries	84.70%	Re-query	80.75%

4 EXPERIMENTS

4.1 Datasets

We experiment with our model and baselines on two real-world datasets: AOL search log [26] and a search log from a commercial search engine (COM). The statistics of the two datasets are shown in Table (1). Besides basic statistics, we calculate "Co-query" and "Re-query" to verify whether short- and long-term interest exists in our dataset. "Co-query" represents the number of two queries that co-occur at least twice within a time window of five queries. It can reflect the short-term local correlations between queries. In the meanwhile, "Re-query" means the percentage of users that have re-query behavior. The statistics of the two metrics show that there indeed exists short- and long-term interest phenomenon in the search behavior of users.

The AOL search log [26] is a popular and publicly available dataset, which contains a query log of three months. Each record

Table 2: Overall performances of baseline and PSTIE on two datasets: AOL and commercial query log. "†" indicates the model outperforms all baseline models significantly with paired t-test at $p < 0.05$ level. The best results are shown in bold.

Models	AOL Query Log						Commercial Query Log					
	MAP	MRR	P-Imp	P@1	P@3	P@5	MAP	MRR	P-Imp	P@1	P@3	P@5
Ori.R	.2529	.2640	-	.1531	.2769	.3492	.7399	.7506	-	.6162	.8459	.9394
K-NRM	.4298	.4399	.6633	.2718	.5130	.6089	.4927	.5007	.0665	.2855	.3391	.3646
MV-LSTM	.4315	.4452	.6605	.2762	.5186	.6131	.4893	.4966	.0624	.2816	.3348	.3592
P-Click	.4221	.4305	.1657	.3780	.4128	.4431	.7509	.7634	.0611	.6260	.8823	.9598
SLTB	.5113	.5237	.3374	.4693	.5244	.5507	.7921	.7998	.1184	.6901	.9016	.9573
HRNN	.5438	.5565	.5927	.4841	.5663	.6042	.8065	.8191	.2401	.7127	.9061	.9590
PSGAN	.5482	.5609	.5985	.4898	.5741	.6190	.8135	.8234	.2494	.7174	.9114	.9658
HRNN+time	.5452	.5554	.5934	.4861	.5623	.6076	.8017	.8136	.2324	.7097	.9012	.9526
HTime-LSTM	.5476	.5578	.5975	.4896	.5677	.6097	.8077	.8210	.2413	.7156	.9131	.9610
H-CTLSTM	.5479	.5574	.5984	.4875	.5687	.6127	.8094	.8231	.2386	.7199	.9165	.9645
PSTIE-REP	.5506	.5610	.6042	.4929	.5734	.6261	.8105	.8238	.2445	.7210	.9181	.9680
PSTIE-ITE	**.5639**†	**.5769**†	**.6847**†	**.5033**†	**.5965**†	**.6413**†	**.8211**†	**.8301**†	**.2636**†	**.7295**†	**.9274**†	**.9766**†

contains an anonymous user ID, a query string, query issued time, a clicked document title, a clicked URL and its original ranking. For each query, we need both clicked and unclicked documents to train our model in a pair-wise way, while the AOL dataset only records the clicked documents. Following [1, 2], we collect the candidate documents for each query from the top documents ranked by BM25 algorithm to solve the above problem. Besides, we keep the background set to provide historical search sequences of each user to build their search interest for the personalized task. We use the first five weeks as the background set to track the search histories of users, the next six weeks for training, and the last two weeks for valid and test set. We sample 5 candidate documents per query for training and validating, and 50 candidates per query in the test set. Following [2, 13, 18], we simply use the document title to measure the relevance of each document.

The commercial (COM) search log was collected from a large-scale commercial search engine from January to February in 2013 [14, 21]. Each record contains a user ID, a query string, query issued time, the top 20 retrieved URLs, click labels and dwell time. This dataset fully keeps the histories of each user because the users are sampled in the dataset construction processing period [21]. Besides, the candidate documents are fully recorded for each query in this dataset, which provides text body for each document. Following [14], we select documents with a dwell time of longer than 30 seconds as clicked documents. We use the first three-quarters of data for background set, and divide the last quarter of the data into the training set, validation set and test set in a 4:1:1 ratio.

4.2 Baselines

We evaluate the performance of our approach by comparing it with several baseline methods. Our baseline models include several traditional ad-hoc search models and state-of-the-art personalized models. We select several RNN-based personalized models to verify the advantage of our model that leverages time information. Besides, we also include several state-of-the-art models that exploit the time information into sequential modeling to prove the effectiveness of our model. Our baseline models include:

K-NRM [10]: This is an ad-hoc interactive matching model, which uses word-level translation matrix and kernel pooling to extract soft match features for query and document.

MV-LSTM [34]: It is an ad-hoc interactive model that matches query and document with multiple positional sentence representations and k-Max pooling to extract the strongest match features.

P-Click [11]: It is a widely used personalized model that counts the number of clicks of documents to measure relevance between query and search history straightly.

SLTB [5]: It is the state-of-the-art traditional personalized method that summarizes features including click-based features, topic features and short- and long-term features by learning to rank.

HRNN [14]: It is a personalized neural model that uses hierarchical RNN and query-aware attention with for modeling dynamic user profiles. It is a suitable baseline because of its clear structure and good scalability to involve time information.

PSGAN [21]: It is the state-of-the-art personalized approach that applies generative adversarial networks to improve the ranking quality. We choose the discriminator and document selection based approach in [21] as our baseline.

HRNN+time: As a naive approach to model time information, we add time interval between each query and each session into the input of HRNN [14] as an additional feature.

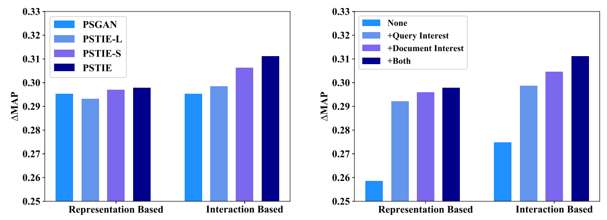
HTime-LSTM: Time-LSTM [40] equips LSTM with time gates to model time intervals. We change the GRU cell in [14] to the third architecture of Time-LSTM cell to exploit the time information.

H-CTLSTM: Time-aware LSTM [23] models the self-exciting dynamics of events in continuous time. We change the GRU cell in [14] to time-aware LSTM cell to exploit time information.

4.3 Settings and Evaluation Metrics

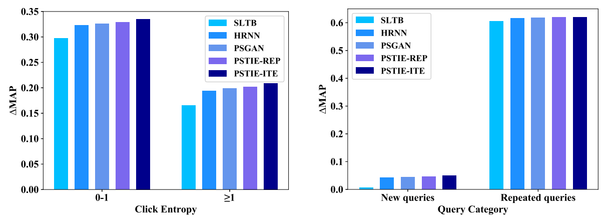
We train the following two proposed models to verify the personalized framework to leverage time information. We use the same parameter set and experiment settings for the two models:

PSTIE-REP: We calculate the personalized score of a candidate document by the query, document, interest, and feature **representations** we build in our model as we illustrated in Section 3.5.1.



(a) long- and short-term interest(b) query and document interest

Figure 3: The effectiveness of interest modeling in PSTIE



(a) Navigational/informational (b) Repeated/new queries

Figure 4: Experiments with different types of queries

PSTIE-ITE: We use time-sensitive interest representations to initialize the query and document positional encoding LSTM, then calculate the final personalized score by the **interactive** component output as proposed in Section 3.5.2. We combine the interactive score and feature score as the final score of a document.

For all datasets, we use 256 as the hidden state of time-aware LSTM, $1e^{-3}$ as the learning rate, k=10 for k-max pooling layer in MV-LSTM, 300 for training batch size. We use Adam Optimizer to train our model. For the AOL dataset, the pre-trained word embedding dimension is 50. For the commercial search engine dataset, the pre-trained word embedding dimension is 300.

We evaluate the results generated by baseline models and our proposed model using Mean Average Precision(MAP), Mean Reciprocal Rank(MRR), Precision in the top k position(P@K). For P@K, we will report P@1, P@3 and P@5 in our experiment. Besides, for measuring more creditable performance on inverse document pairs [19], we also compute the percentage of improved pairs(P-Improve) following previous personalized studies [14, 21].

4.4 Overall Performance

We compare PSTIE that we proposed in our paper with the baselines on two datasets, and present the result in Table 2. We find the results are entirely consistent among the datasets. We have obtained observations as follows:

(1) **Our PSTIE approach outperforms all baseline methods, including ad-hoc models, personalized models, and time information enhanced models.** Compared with the state-of-the-art model PSGAN on the AOL dataset, The MAP has been improved 4.4‰ by PSTIE-REP and 28.6‰ by PSTIE-ITE, proving that leveraging time information can help to build more accurate user interest for personalized search. Besides, the MAP has been improved 4.9‰ by PSTIE-REP and 29.2‰ by PSTIE-ITE compared to H-CTLSTM that replaces the cell in HRNN with time-aware LSTM cell. The MAP improvement shows that we can exploit time information more effectively by considering the special interest evolution and long-term re-finding behavior. Our models on the commercial dataset show less improvement than AOL. One possible reason can be that the original ranking of the commercial dataset is already of high quality, leaving little room for our model to improve.

(2) The interaction based model PSTIE-ITE outperforms baseline models with a significant improvement, which validates the effectiveness of our model. Compared to PS-ITE, the representation based model PS-REP only achieves comparable results with state-of-the-art model PSGAN, which indicates that interactive methods considering the matching features between each word of query and document can be more effective than representation-based methods. However, we find the performance of PSTIE-REP is more stable than PSTIE-ITE in multiple trials, indicating that representation-based methods can retain more interest information of users than interactive methods. Besides, the model we propose can be ensembled with other models such as PSGAN to achieve better results. The improvement of the two methods compared to baselines shows that each of them can fully leverage the time information to capture the long- and short-term user interests for personalized search.

(3) Ad-hoc methods, including MV-LSTM and K-NRM, can improve the results significantly on the AOL log but show poor performance on the commercial log. One possible reason is that the original ranking list generated based on BM-25 is quite naive in the AOL log, leaving enough improvement room for K-NRM and MV-LSTM to re-rank the documents by extracting document and query matching features. The personalized methods (e.g., P-Click, HRNN, PSTIE), which use the history of users to re-rank the document list, can significantly outperform the original ranking result and ad-hoc methods without a search history. Neural personalized methods (e.g., HRNN, PSGAN, PSTIE) can dramatically outperform the feature-based techniques (e.g., P-Click, SLTB), proving the power of deep learning to capture sequential patterns and user profiles.

(4) The methods that consider time information(e.g., HTime-LSTM, PSTIE) perform better than HRNN, showing the effectiveness of time information for building more accurate user interests. HTime-LSTM equips LSTM with time gates to model the time intervals, while H-CTLSTM models the interest of users in continuous time with a decay rate over time. Both models perform better than HRNN and naive HRNN+time, but have some limitations: they cannot capture the long-term re-finding behavior of users effectively. Therefore, it yields worse performance than our proposed PSTIE.

In summary, the results show that **leveraging the time information by considering the short-term time-sensitive interest evolution and long-term re-finding behavior can help build more accurate interest of users than previous neural sequential models**. We will further analyze the effectiveness of short-term and long-term interests next.

4.5 Long- and Short-Term User Interest

In this section, we conduct several experiments to explore the effectiveness of learning the long- and short-term user interest. We compare the performance of our methods with only the long-term user interest construction model PSTIE-L and the short-term

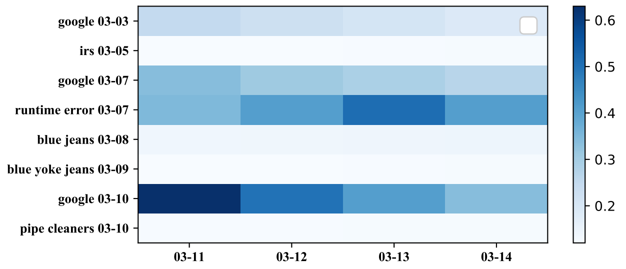


Figure 5: Visualization of change of long-term interest

user interest construction model PSTIE-S. For each model, both the query intent and document interest are included. We report the performance of each model on the AOL log, evaluated on ΔMAP.

As shown in Figure 3(a), we can find that all of our proposed models outperform baseline model PSGAN, demonstrating the effectiveness of our methods leverage time information. The PSTIE-S performs better than PSTIE-L on both representation- and interaction-based methods, proving that short-term interest and local sequential patterns contribute more to personalized search. One possible reason can be that many queries have not been re-queried, and many documents have a small number of clicks, which long-term re-finding interest cannot fit. Besides, combining PSTIE-S and PSTIE-L using both short- and long-term user interest can effectively improve the personalized performance, validating that incorporating short- and long-term can be useful to construct user interest more accurately and is beneficial to personalized search.

4.6 Query Intent and Document Interest

We conduct experiments to validate the effectiveness of query intent and document interest in this section. We compare the performance of our approach with its variants without query intents or/and document interest on the AOL log. In the representation-based method, the method "None" only uses relevance features for re-ranking. In the interaction-based method, it initializes bi-directional LSTM with random states by the uniform distribution.

According to Figure 3(b), incorporating either query intent or document interest can effectively improve the performance of our approach. Besides, the methods with document interest outperform the methods with query intent in both situations. One possible reason is that the clicked document can reflect the preference and interest of users more straightly and thoroughly, while the issued query may deviate from his real intent [21]. Moreover, we find that combining query intent and document interest perform the best, validating the effectiveness of our approach in exploiting both query and document history sequences for constructing user interests.

4.7 Different Types of Queries

We conduct experiments to validate the effectiveness of our model on different queries and situations in the AOL log. We design two situations of different queries to experiment on.

4.7.1 Navigational queries and Informational queries. Queries of users can be divided info navigational queries and informational queries based on its click entropy [11, 14]. Navigational queries represent queries with clear intent, while informational queries can be ambiguous and have more rich semantics. Informative queries

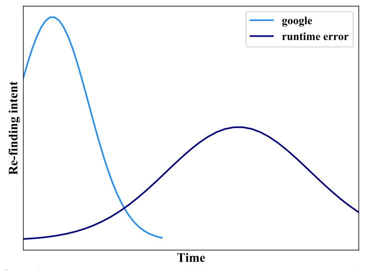


Figure 6: Re-finding intent Gaussian mixture distribution of query "google" and "runtime error"

tend to leave more improvement space for personalization since different users can have various preferences behind the same query. Based on that, we divide the queries into two groups with the cutoff of click entropy at 1.0. We compare the personalized performance of our model and baseline models on these two groups of queries.

As shown in Figure 4(a), our methods can outperform all baseline models on both two query sets. However, differently from the previous work [14, 21], The performance of all models on navigational queries is better than that on informational queries on AOL log. One possible reason can be that the original ranking of the AOL dataset is based on BM25, whose quality is lower than that in commercial search engine log. However, the improvement of our model on both groups of queries is significant, validating the ability of PSTIE to tailor the personalized interests of users with fine-grained time information to improve ranking performance.

4.7.2 Repeated queries and New queries. Users usually issue similar queries in their search history, showing re-finding behavior for personalized models to learn and improve the ranking results. Thus, we divide the queries of users in the test set into two groups: repeated queries and new queries. We experiment with our model on the two groups of queries and summarize the result in Figure 4(b).

According to Figure 4(b), all experimented personalized methods can improve the ranking results significantly on repeated queries. Our PSTIE model can outperform all baseline models, proving the effectiveness of our methods to exploit the re-finding behavior of users in their search history. Besides, our PSTIE-L methods with only long-term re-finding interest outperform PSGAN and PSTIE-S significantly, while PSTIE-S only shows comparable results with PSGAN. This observation indicates that the long-term interest in PSTIE can indeed capture the time-sensitive re-finding behavior of users. PSTIE-ITE also shows significant improvement in non-repeated queries compared to baseline models, proving the effectiveness of the interactive matching component to extract matching features of new queries and documents to measure user preference.

4.8 Interpretability and Case Study

Users tend to re-find some topics or documents at the end of the lifetime of information. To leverage re-finding behavior, we design a query-specific Gaussian mixture distribution to measure the re-finding interest tendency of users in Section 3.4. In Equation (10), the mean μ_i can be seen as the expected lifetime of one query, while the coefficient π_i can be the importance of this re-finding effect. Thus, the model we propose provides strong interpretability.

To further analyze the influence of the long-term interest we build, we present an example by sampling one user with a relatively long history in the AOL dataset and visualize the re-finding tendency α_i calculated by Equation (10) of each query. We remove the queries that have weights less than 0.01 to highlight important ones. We choose four issued times as the current time. As shown in Figure 5, the results approximate real contributions and our expectations. Specifically, the query "google" is a frequently issued query, which should be estimated at a smaller μ as its lifetime and larger importance π. In the meanwhile, "Runtime Error" should have a longer life as a non-daily query with smaller importance. We can find that at "03-11", the query "google" at "03-10" has the most significant weight while the influence of other queries is small. However, at "03-13", the historical query "runtime error" at "03-07" shows comparable weight as the query "google". We further demonstrate the Gaussian mixture distribution of query "google" and "runtime error" in Figure 6. We can find that the query "runtime error" tends to have a lifetime around one week since that of google is less than one day, while the latter has a larger coefficient than the former one, which corresponds to the expectations.

The visualization indicates that our model can capture the re-finding behavior of users effectively and utilize it to enhance personalized performance. Besides, it shows strong interpretability to explain re-finding tendency drifting with time.

5 CONCLUSION

In this paper, we propose PSTIE to leverage the fine-grained time information to construct more accurate interest representations of users for personalized search. We design two time-aware LSTM architectures to model the time-decaying evolution of query intent and document interest, then build short-term interest to represent the local preference of the user. We further model the lifetime re-finding interest of the user with query-specific Gaussian mixture distribution drifting with time, then build the long-term user interest. Besides, we propose two methods to fuse long- and short-term user interest to re-rank the document list, i.e., representation-based method by using the cosine similarity, or interactive-based method by initializing the Bi-directional LSTM with user interest in query and document matching. Extensive experiments on two real-world datasets show our approach can effectively improve the performance of personalized ranking. In the future, we plan to exploit the clicked time information and dwell time of each document to model more fine-grained time-sensitive behavior of users.

ACKNOWLEDGMENTS

Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 61872370 and No. 61832017, and Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098.

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-Task Learning for Document Ranking and Query Suggestion. In *ICLR 2018*.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. In *SIGIR 2019*. ACM, 385–394.

[3] Ting Bai, Lixin Zou, Wayne Xin Zhao, Pan Du, Weidong Liu, Jian-Yun Nie, and Ji-Rong Wen. 2019. CTRec: A Long-Short Demands Evolution Model for Continuous-Time Recommendation. In *Proceedings of SIGIR 2019*. ACM, 675–684.

[4] Paul N. Bennett, Krysta M. Svore, and Susan T. Dumais. 2010. Classification-enhanced ranking. In *Proceedings of the WWW 2010, 2010*. 111–120.

[5] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the SIGIR 2012*. 185–194.

[6] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of ICML 2005, 2005*. 89–96.

[7] Fei Cai, Shangsong Liang, and Maarten de Rijke. 2014. Personalized document re-ranking based on Bayesian probabilistic matrix factorization. In *Proceedings of the SIGIR '2014*. ACM, 835–838.

[8] Mark James Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards query log based personalization using topic models. In *Proceedings of the CIKM 2010*. 1849–1852.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the DLRS@RecSys 2016*. 7–10.

[10] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proceedings of the WSDM 2018*. 126–134.

[11] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW 2007*. 581–590.

[12] A S Fotheringham and D W S Wong. 1991. The Modifiable Areal Unit Problem in Multivariate Statistical Analysis. *Environment and Planning A: Economy and Space* 23, 7 (1991), 1025–1044. https://doi.org/10.1068/a231025

[13] Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of the CIKM 2010*. ACM, 1139–1148.

[14] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing Search Results Using Hierarchical RNN with Query-aware Attention. In *Proceedings of the CIKM 2018*. 347–356.

[15] Morgan Harvey, Fabio Crestani, and Mark James Carman. 2013. Building user profiles from topic models for personalised search. In *CIKM'13*. 2309–2314.

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

[17] Seyyed Abbas Hosseini, Keivan Alizadeh, Ali Khodadadi, et al. 2017. Recurrent Poisson Factorization for Temporal Recommendation. In *SIGKDD 2017*. 847–855.

[18] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of CIKM' 2013*. 2333–2338.

[19] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR 2005*. 154–161.

[20] Xiujun Li, Chenlei Guo, Wei Chu, Ye-Yi Wang, and Jude Shavlik. 2016. Deep Learning Powered In-Session Contextual Ranking using Clickthrough Data.

[21] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. PSGAN: A Minimax Game for Personalized Search with Limited and Noisy Click Data. In *Proceedings of the SIGIR 2019*.

[22] Nicolaas Matthijs and Filip Radlinski. 2011. Personalizing web search using long term browsing history. In *Proceedings of the WSDM 2011*. 25–34.

[23] Hongyuan Mei and Jason Eisner. 2017. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *NIPS 2017*. 6754–6764.

[24] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md. Mustafizur Rahman, Pinar Karagoz, , et al. 2018. Neural information retrieval: at the end of the early years. *Inf. Retr. Journal* 21, 2-3 (2018), 111–182.

[25] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, et al. 2016. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM TASLP* 24, 4 (2016), 694–707.

[26] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the Infoscale 2006 (ACM)*, Vol. 152. ACM, 1.

[27] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the EMNLP 2014*.

[28] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the CIKM 2014*. 101–110.

[29] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. 2007. Web search personalization with ontological user profiles. In *Proceedings of the CIKM 2007*. 525–534.

[30] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep RankNet for personalized search. In *Proceedings of the WSDM 2014*. 83–92.

[31] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael A. S. Potts. 2007. Information re-retrieval: repeat queries in Yahoo's logs. In *SIGIR 2007*. 151–158.

[32] Sarah K. Tyler, Jian Wang, and Yi Zhang. 2010. Utilizing re-finding for personalized information retrieval. In *Proceedings of the CIKM 2010*. 1469–1472.

[33] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search Personalization with Embeddings. In *ECIR 2017, Proceedings*.

[34] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. In *Proceedings of the AAAI 2016*. 2835–2841.

[35] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling Item-Specific Temporal Dynamics of Repeat Consumption for Recommender Systems. In *The World Wide Web Conference, WWW 2019*. 1977–1987.

[36] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR 2017*. 55–64.

[37] Jing Yao, Zhicheng Dou, Jun Xu, and Ji-Rong Wen. 2020. RLPer: A Reinforcement Learning Model for Personalized Search. In *WWW 2020*. ACM, 2298–2308.

[38] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, et al. 2019. Where to Go Next: A Spatio-Temporal Gated Network for Next POI Recommendation. In *AAAI 2019*. 5877–5884.

[39] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Enhancing Re-finding Behavior with External Memories for Personalized Search. In *WSDM 2020*. ACM.

[40] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM. In *Proceedings of the IJCAI 2017*. 3602–3608.