Encoding History with Context-aware Representation Learning for Personalized Search

Yujia Zhou[2], Zhicheng Dou[1], 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
zhouyujia@ruc.edu.cn,dou@ruc.edu.cn,jirong.wen@gmail.com

ABSTRACT

The key to personalized search is to clarify the meaning of current query based on user's search history. Previous personalized studies tried to build user profiles on the basis of historical data to tailor the ranking. However, we argue that the user profile based methods do not really disambiguate the current query. They still retain some semantic bias when building user profiles. In this paper, we propose to encode history with context-aware representation learning to enhance the representation of current query, which is a direct way to clarify the user's information need. Specifically, endowed with the benefit from transformer on aggregating contextual information, we devise a query disambiguation model to parse the meaning of current query in multiple stages. Moreover, for covering the cases that current query is not sufficient to express the intent, we train a personalized language model to predict user intent from existing queries. Under the interaction of two sub-models, we can generate the context-aware representation of current query and re-rank the results based on it. Experimental results show the significant improvement of our model compared with previous methods.

CCS CONCEPTS

• **Information systems → Personalization**.

KEYWORDS

Personalized search; Context-aware model; Hierarchical transformer

ACM Reference Format:
Yujia Zhou, Zhicheng Dou, Ji-Rong Wen. 2020. Encoding History with Context-aware Representation Learning for Personalized Search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401175

1 INTRODUCTION

Search engine has become a common tool for obtaining information from the web. Due to differences in individual preferences, the same query often represents different query intents. Therefore, it is difficult to meet the needs of all users by returning the same document list for them. In order to solve this problem, personalized search tries to model the user's preference and re-rank the search results based on it. Traditional strategies of personalized search [4, 6, 13, 29, 31, 35, 36] built user models by extracting personalized features from the query logs. They prove the value of user historical click-through data to tailor the ranking. In recent years, deep learning based methods [12, 17, 18, 29] were proposed to learn the representations for discovering deep semantic relevance, which brought significant improvements in retrieval quality.

Previous methods have revealed that the majority of queries issued by users are short and ambiguous [8, 28]. Capturing user's real intent is a critical step for personalization when receiving an ambiguous query. Most of existing studies focus on building user profiles according to user historical data, and highlighted the relevant part based on the current query to re-rank the candidate documents. However, we argue that they did not essentially disambiguate the current query, but summarize user interests from historical behaviors. This will still cause semantic deviations when building the user profiles due to the ambiguity of the current query. For example, assuming that a user searched the query "java programming" and "Hawaii Island" in the past. For the current query "java", which combines the semantics of "java programming language" and "java island", user profile based methods will highlight both of historical queries and consider their impact on the results. Such a user profile is biased owing to the ambiguity of "java". Intuitively, in specific context of user's search history, the semantic representation of "java" should be unique and reflects clear information needs. In this paper, we aim to abandon the establishment of user profile. Instead we regard the historical data as contextual information to refine the representation of the current query.

Existing context-aware methods of document ranking are mainly to encode the contextual information through Recurrent Neural Networks [2, 20] or Convolutional Neural Network [9]. However, these methods focus on local relations which still face the challenge of long-term dependency. Transformer [32], a structure based on self-attention mechanism, has achieved great success in various tasks thanks to the characteristic of global interaction. Its advantages perfectly fit the needs of personalized search on long-term dependency. Due to its powerful ability of leveraging contextual information, we intend to apply it to encode history.

The ambiguity of the query mainly comes from vague terms in it, which may lead to misunderstanding. And making use of the context of surrounding terms is a feasible solution to disambiguation.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401175

However, since queries are often short and consist of only a few keywords, relying on limited words is not enough to deduce user intent. It is necessary to use search history to disambiguate the current query. Previous studies [4, 12, 16] have considered the effect of historical data in different time period. They divided the search history into short-term (current session) and long-term (previous sessions) to model user interests respectively. Due to the hierarchical structure of query logs, we intend to establish a query disambiguation model with hierarchical transformer to disambiguate the current query in multiple stages. In addition, due to the randomness of user's behavior when searching, the query sometimes is hard to express the information needs as a result of misspelling or deviating expression. In such a case, user's real intent cannot be obtained simply by disambiguation. Inspired by the masked language model in BERT [10], we believe that the series of queries for an information need have a specific pattern. We attempt to build a personalized language model capturing such patterns to predict the user intent.

Specifically, we propose a context-aware neural retrieval model, which incorporates rich contextual information including query terms, short-term history, and long-term history to refine the representation of the current query. It consists of two hierarchical transformer-based sub-models, the query disambiguation model and the personalized language model. The former is used to disambiguate the current query multiple times on the basis of its terms and historical interactions, while the latter tries to learn the personalized search patterns of users to more accurately infer the user intent. Finally, with the interaction of the two sub-models, we can obtain a context-aware query representation with specific intent, and compute its relevance to each candidate documents to personalize the results.

The main contributions of this paper are summarized as follows: (1) We encode the user's history with context-aware representation learning for personalized search without user profile. (2) We devise a query disambiguation model with rich contextual information to understand the user needs on the basis of the current query. (3) To cover situations where the current query fails to express information needs, we build a personalized language model to predict the user intent according to existing queries. (4) We apply a gate to fuse the two sub-models, and add supervision information to the predicted user intent for further optimization.

The rest of paper is arranged as follows. Related works are summarized in Section 2. The proposed method is shown in Section 3. We introduce the experimental settings in Section 4, and analyze the results in Section 5. The conclusion is drawn in Section 6.

2 RELATED WORK

2.1 Personalizing Web Search

Personalized search has become a hot research field due to its ability to meet the information needs of different users, and it has been proven to effectively improve the quality of ranking [6]. The key to personalization is how to accurately capture user preferences by analyzing user historical queries and click information. Therefore, a large number of methods for mining personalized information from user query logs have been proposed.

Some early studies focused on the click-based features contained in the query logs, which is accessible and reliable for predicting user preferences. Dou et al. [11] proposed the P-Click model to predict the probability of clicking by counting the number of historical clicks. The similar approach was used in [31] to figure out the personal navigation for personalizing search results. In addition, some studies attempt to analyze the topic features of documents to build user models. The Open Directory Project (ODP) was widely used to represent the topic of a web page [3, 27, 37]. Unfortunately, this might lead to huge labor costs and incomplete categories. Later works [7, 13, 34, 36] tried to learn the topic representations automatically and learned a latent user preference vector, which showed superiority over previous methods. In particular, most of these studies point out that personalization strategies are not suitable for all queries, and usually have a distinct advantage on ambiguous queries. Therefore, the click entropy [11] and the topic entropy [30] were proposed to measure the ambiguity of a query. The emergence of the learning to rank method allows us to combine multiple personalized features in a non-linear manner. Previous studies [4, 33, 37] made great success in training ranking model with the advanced ranking algorithm LambdaMART [39].

Deep learning is widely used in various fields due to its powerful representation learning capabilities. For personalized search, it is a great tool for discovering potential user preferences. Song et al. [29] leveraged individual information to adapt the general ranking model. Li et al. [17] expected to improve the results with the employment of semantic features of in-session contexts. Ge et al. [12] proposed hierarchical recurrent neural networks with query-aware attention to model the sequential information and to eliminate the irrelevant interests. Lu et al. [18] proposed PSGAN framework, which is based on generative adversarial network to generate high quality negative examples for training. Different from these user profile based studies, we focus on encoding user's search history as context to enhance the query representation.

2.2 Context-aware document ranking

Recently, context-aware language models have made impressive progress on various natural language processing tasks, such as ELMo [24] and BERT [10]. These studies have proven to be effective in representing sentences without ambiguity, which is a key point of the document ranking task. Prior work has indicated that combining contextual information can better represent queries and documents for ranking. Dai et al. [9] devised conv-KNRM model, using convolutional neural network to learn context-aware word representations. Mcdonald et al. [20] applied the recurrent neural network to model sequential information in query terms. Ahmad et al. [2] presented the CARS, a context attentive document ranking and query suggestion model. They leveraged recurrent neural network to encode the contextual information and to exploit users' on-task search activities. Due to the stronger ability of the transformer [32] to extract latent features, more transformer-based methods were proposed to aggregate context. [22, 41, 42] used BERT sentence classification to predict the relevance of query-document pairs. MacAvaney et al. [19] combined the existing neural ranking model with BERT's term representations and achieved higher precision. Similarly, we intend to learn the context-aware query representation with transformer for personalized search.

3 HTPS: THE PROPOSED MODEL

Personalized search plays an important role in capturing users' real intents, which can potentially improve the search results to meet individuals' needs. As we stated in Section 1, most of existing personalized methods tailor the results based on building user profiles. This way of modeling user interests still retains the static representation of the current query, which might be ambiguous. In this paper, we propose to learn the context-aware representation of the current query, which can be regarded as user's real intent directly. Specifically, endowed with rich contextual information hidden in query terms, short-term history and long-term history, a query disambiguation model is constructed to parse the meaning of current query in multiple stages. Furthermore, in order to cover the case of misspelling and misrepresenting, we build a personalized language model to help understand user intents.

To start with, the problem can be defined as follows. Suppose that for a user, his historical data H consists of the short-term history H^s and the long-term history H^l. The former includes a series of queries and candidate documents in the current session, $H^s = \{\{q_1^s, D_1^s\}, ..., \{q_{t-1}^s, D_{t-1}^s\}\}$, where t is the current timestamp. The latter contains user past interactions in previous sessions, $H^l = \{\{q_1^l, D_1^l\}, ..., \{q_n^l, D_n^l\}\}$, where n is the number of queries issued in previous sessions. Given a new query q and candidate documents $D = \{d_1, d_2, ...\}$ returned by the search engine, our task is to score each element in D based on the current query q and the historical data H. The score of the document d is denoted as $p(d|q, H)$.

Different from previous user profile based methods, which focus on extracting personalized features from the historical data H, we attempt to enhance the representation of the current query q based on it. The personalized search results are generated according to the matching scores of refined query representation q^H and each candidate document. The final score can be computed as:

$$p(d|q, H) = \phi(p(d, q), p(d, q^H)), \tag{1}$$

where $p(d, q)$ is the adhoc relevance between the document and the query. And $p(d, q^H)$ represents the personalized relevance with respect to the context-aware query representation enhanced by the historical data. The function $\phi(\cdot)$ is the multilayer perceptron (MLP) with $tanh(\cdot)$ as the activation function, which is used to combine the score of each part with different weights. In the remaining of the section, we will introduce the refining process in detail.

3.1 Multi-stage Query Disambiguation

As we discussed in Section 1, a large part of queries user issued are ambiguous. This hinders us from understanding the real intents of users and returning a list of documents they are satisfied with. To solve this problem intuitively, we intend to build a multi-stage query disambiguation model which integrates the query terms, short-term history, and long-term history respectively. A hierarchical transformer structure is applied to encode them as context for clarifying the meaning of current query. Specifically, we divide the whole process into (1) word-level disambiguation, which considers the effect of surrounding terms; (2) short-term disambiguation, which regards the user interactions in short-term history as the context; (3) long-term disambiguation, which uses the context of user interactions in long-term history. The structure of the query disambiguation model is shown in Figure 1. We will elaborate on the role of each stage and how to achieve it in the following.

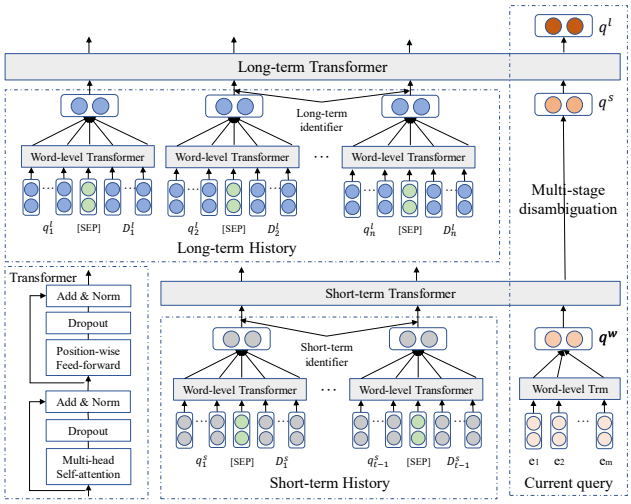


Figure 1: The architecture of the query disambiguation model. Given a new query, we encode the query terms, short-term history, and long-term history as contextual information separately with hierarchical transformer to disambiguate the current query .

3.1.1 Word-level disambiguation. For vague words in ambiguous queries, the context of surrounding words provides an opportunity to capture the true meaning of them and further understand the query intent. For example, the word "apple" has different meanings in the query "apple fruit" and "apple company" because of different context. Instead of pre-trained word vectors, we hold the idea that even the same word should be represented by different vectors in different contexts.

For the current query q, suppose it consists of m terms, denoted as $q = \{e_1, e_2, ..., e_m\}$. We aim to learn the context-aware representation of each term with the word-level transformer based on the entire text, denoted as:

$$E^w = \text{Trm}(q + q_p), \tag{2}$$

where $q \in \mathbb{R}^{m*d}$ and $q_p \in \mathbb{R}^{m*d}$ are the word embeddings and the position embeddings of the terms in query q. And $\text{Trm}(\cdot)$ is a transformer layer following [32], containing a Multi-head Self-attention (MS) layer and a Position-wise Feed-forward (PF) layer. To prevent difficult training as the network goes deeper, we apply residual connections to each layer.

$$\text{Trm}(q) = \text{LN}(M_q + \text{D}(\text{PF}(M_q))),$$
$$M_q = \text{LN}(q + \text{D}(\text{MS}(q))),$$

where $\text{LN}(\cdot)$ is layer normalization to stabilize the output. And $\text{D}(\cdot)$ is a dropout layer with 0.1 probability in our settings. The multi-head self-attention has been shown to perform better than traditional attention mechanism due to its ability to apply weights with different focuses [32]. Specifically, it projects the inputs into h subspaces with different parameters firstly, and employ the single attention (Att) for each head. The final output is generated by

concatenating them.

$$\text{MS}(q) = [head_1, ..., head_h]W^O,$$
$$head_i = \text{Att}(qW_i^Q, qW_i^K, qW_i^V),$$
$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d/h}}\right)V,$$

where the projection matrices of each head $W_i^Q \in \mathbb{R}^{d*d/h}$, $W_i^K \in \mathbb{R}^{d*d/h}$, $W_i^V \in \mathbb{R}^{d*d/h}$, and $W^O \in \mathbb{R}^{d*d}$ are parameters learned during the training. To endow the model with interactions between different dimensions, we add a position-wise feed-forward network to enhance the representation with nonlinear projections. It consists of two convolutions with kernel size 1. We have:

$$\text{PN}(x) = \text{C}_2(\text{ReLU}(\text{C}_1(x^T)))^T,$$

where $\text{C}_1(\cdot)$ and $\text{C}_2(\cdot)$ are two convolutions with different parameters, and they are shared across all positions. Finally, the outputs of transformer layer in Eq. (2) are viewed as the context-aware representations of terms in the current query, i.e. $E^w = \{e_1^w, e_2^w, ..., e_m^w\}$. To reduce downstream computational costs, we represent the query with word-level disambiguation q^w by summing its terms, i.e.,

$$q^w = \sum_{i=1}^{m} e_i^w.$$

For cover more cases, we will continue to disambiguate the current query with q^w as the input for the next stage.

3.1.2 Short-term disambiguation. As we have discussed in Section 1, the queries issued by users are often very short, even one word. In such a case, word-level disambiguation can not completely eliminate the polysemy. There is a common scenario that users often present a series of queries in a session for a single information need. The queries and click data during this process provide rich context information to deduce the current user intent. Therefore, we intend to further disambiguate the current query by incorporating the context of short-term history.

Formally, for each query in short-term history H^s, we join the query terms with its satisfied documents, with "[SEP]" as the separator. After the word-level disambiguation, we take the output $h_i^{s\,w}$ as contextual information for the current query. All of the outputs form the short-term context, denoted as $H^{s\,w} = \{h_1^{s\,w}, ..., h_{t-1}^{s\,w}\}$. We concatenate $H^{s\,w}$ with q^w and apply the short-term transformer to get the refined representation q^s.

$$q^s = \text{Trm}^{last}([H^{s\,w}, q^w] + [H^{s\,w}, q^w]_{p^+}),$$

where $\text{Trm}^{last}(\cdot)$ means merely taking the output of the last position, which corresponds to the current query. The position embedding p^+ here not only encodes the relative position in the sequence, but also considers the hierarchical information. The basic idea is that the short-term history is closer in absolute position and should be paid more attention to than the long-term history. Thus, we add an identifier to distinguish whether the current context belongs to short-term (marked as 2) or long-term history (marked as 1), and embed it in the same way as position embedding. The output q^s will be further refined in the next stage.

3.1.3 Long-term disambiguation. After the above two steps, the intents of some queries has been accurately represented. But for the short queries at the beginning of the session, they are still biased for lack of context. Long-term history often reflects users' stable and solidified interests, which also provide a way to infer user intents based on the current query. Thus, we encode the long-term history as context for further disambiguation. Similar to the short-term disambiguation, after word-level disambiguation for each item in H^l, we put the output vectors $H^{l\,w} = \{h_1^{l\,w}, ..., h_n^{l\,w}\}$ and q^s together as the input of the long-term transformer. And the output of the last position q^l is considered as the new query representation enhanced by the long-term history, computed as:

$$q^l = \text{Trm}^{last}([H^{l\,w}, q^s] + [H^{l\,w}, q^s]_{p^+}).$$

Finally, we have encoded the history as context to disambiguate the current query multiple times. The context-aware representation at each stage q^w, q^s, and q^l will contribute to the final matching. However, they are not enough to accurately express the user intent in some cases. We will introduce the limitations of the query disambiguation model and how to get over it in the next part.

3.2 Personalized language model

The premise that the query disambiguation model we proposed above can work is that the user's real intent is hidden in the current query, and we can strip it out step by step. However, sometimes there is a deviation between the issued query and real information needs. For instance, a user wants to find the detailed information of a medicine (assuming aspirin), but he forgets the name of it. In such a case, he might searched by "a common antiphlogistic medicine" (deviating expression), or a guessed name "asp.." (misspelling). These queries do not contain ambiguous terms and they are difficult to be refined by simple disambiguation. Thus, as shown in Figure 2, we aim to establish a personalized language model to predict the user's real intent.

Looking at the entire query logs, the same search pattern often appears multiple times. In other words, a series of inaccurate queries in a session can also reflect a specific information need, which will help understand the user intent when the search engine meets them again. Our goal is to learn such a search pattern and to predict the user intent based on existing queries. There are two steps in this process which will be introduced as follows.

3.2.1 Modeling the current search pattern. Given the set of issued queries in the current session, $Q^s = \{q_1^s, ..., q_{t-1}^s, q\}$, we intend to use the similar search patterns in the long-term history to enhance the query representations of Q^s. We denote the set of historical queries in long-term history as $Q^l = \{q_1^l, ..., q_n^l\}$, each query is represented by averaging the vector of each word. In particular, to segment the search patterns, we add the "[SEP]" token to session boundaries, and encode the session ID into the query embedding. The series of queries with the same ID can reflect a single information need. Finally, we concatenate the queries in previous sessions Q^l and the queries in current session Q^s, and apply a low-level transformer to model the search pattern of current session.

$$Q^{s\,l} = \text{Trm}([Q^l, Q^s] + [Q^l, Q^s]_p + [Q^l, Q^s]_s).$$

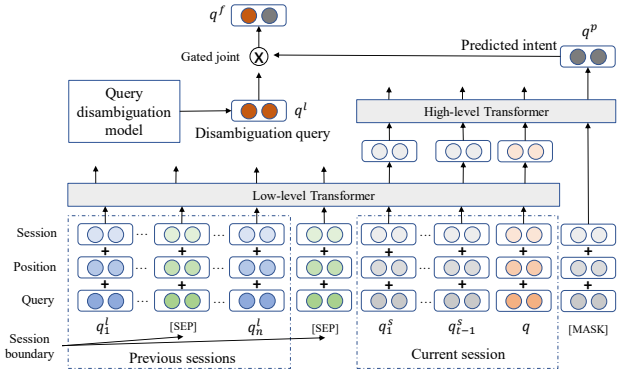


Figure 2: The architecture of the personalized language model, which is used to learn the user's personalized search pattern. We first enhance the queries in the current session based on historical patterns, and then summarize them to predict the current intent. Finally, we combine two models with a gate unit.

where $[Q^l, Q^s]_s$ is the encoding of the session ID. And the output set Q^{s^l} are refined representations of Q^s which incorporates the personalized search patterns. It will contribute to the prediction of user's real intent in the next.

3.2.2 Predicting the real intent. Now that we have enhanced the query representations of the current session based on the user's personalized search patterns, we attempt to predict the real intent based on them with a high-level transformer. Specifically, we add a "[MASK]" token at the last position in the sequence Q^{s^l} and regard its output q^p as the predicted intent.

$$q^p = \text{Trm}^{last}\left(\left[Q^{s^l}, [\text{MASK}]\right] + \left[Q^{s^l}, [\text{MASK}]\right]_p\right).$$

The predicted intent q^p summarizes all existing queries and infers the most likely intent at the current time. For better optimizing this model, we choose the next query to supervise the predictions, which will be demonstrated in Section 3.5.

3.3 Gated Joint of Two Models

The disambiguated representation q^l and the estimated user intent q^p are both significant references for inferring the user's information need. We believe that they contribute differently under different conditions, and their weights should be dynamically adjusted as the issued query changes. In order to automatically learn the contribution of the two sub-models to the final results, we devise a gate unit to control the flow of both parts. Formally, the final representation of the current query q^f is generated by aggregating the q^l and q^p with the gate weight z:

$$q^f = z * q^l + (1 - z) * q^p,$$

where z is learned according to the characteristic of current query and the two parts by MLP. Note that in order to make z between 0 and 1, the activation function here is sigmoid(·).

$$z = \phi\left([q, q^p, q^l]\right).$$

The final representation q^f combines the information of query disambiguation model and personalized language model, and it acts on computing the matching score to re-rank the results.

3.4 Re-ranking Search Results

Finally, the final score of each candidate document can be calculated by the context-aware query representations collected above. For a wider range of matching, we apply the word-level disambiguation to the candidate document and the output vector is denoted as d^w, which is the context-aware representation of d. Next we will introduce the computing method of each part in Eq. (1) in detail.

For the personalized relevance $p(d, q^H)$, we collect the refined query representations enhanced by history at each stage, including (1) the query with short-term and long-term disambiguation q^s and q^l in Section 3.1, (2) the predicted user intent q^p in Section 3.2, and (3) the final representation q^f in Section 3.3. For covering queries with varying degrees of ambiguity semantic deviation, we match them all to the document d^w and automatically adjust the weights for each part using MLP. We have:

$$p(d, q^H) = \phi\left(s^R(q^s, d^w), s^R(q^l, d^w), s^R(q^p, d^w), s^R(q^f, d^w)\right),$$

where $s^R(\cdot)$ means the representation based similarity. It is implemented as the cosine similarity in this paper.

For the adhoc relevance $p(d, q)$, we consider the matching of the original query and document terms, and their refined representations q^w and d^w. Moreover, following [4], we extract several additional features $\mathcal{F}_{q,d}$ for each document about the clicks and topic. These features are fed into MLP to compute a relevant score. Formally, the adhoc relevance consists of three parts:

$$p(d, q) = \phi\left(s^I(q, d), s^R(q^w, d^w), \phi(\mathcal{F}_{q,d})\right),$$

where $s^I(\cdot)$ is the interaction based similarity. We follow the KNRM model proposed by [40] to implement such an interactive match. Given a query-document pair, we construct an interaction matrix M by cosine similarity between each term in query and document. And then we apply k kernels to handle the different ranges of matching. The final matching score $F_k(M)$ is generated by aggregating them with MLP:

$$F_k(M) = \phi\left(f_1(M), f_2(M), ..., f_k(M)\right),$$

$$f_k(M) = \sum_i \log\left(\sum_j \exp\left(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}\right)\right),$$

where μ_k is distributed in -1 and 1 according to the number of kernels, and σ_k is set to 0.1. Finally, by re-ranking the results based on the final relevance score, we obtain the personalized search results with respect to the user's information need.

3.5 Training and Optimization

In this section, we will introduce how to train the model and optimize the parameters. Aside from tuning the ranking model that directly contribute to the personalized results, we devise the loss of the predicted user intent from the personalized language model to further improve the results.

Ranking loss. We adopt the LambdaRank algorithm [5] to train the ranking model in a pairwise manner. Each pair is constructed

Table 1: Basic statistics of the datasets.

Type	AOL dataset	B dataset
#days	91	58
#users	110,439	33,204
#queries	736,454	2,665,625
#sessions	279,930	654,776
Average query length	2.87	3.25
Average session length	2.55	2.63
Average #click per query	1.11	0.46

with a positive sample (clicked document) and a negative sample (unclicked document). Our goal is to maximize the gap between the scores of them. Formally, given a positive document d_i and a negative document d_j, the predicted probability that d_i is more relevant than d_j is computed by $p(d_i|q,H)-p(d_j|q,H)$ with sigmoid normalization. The loss function of ranking model is computed by the weighted cross entropy between the true lable $\overline{p}_{ij}$ and the predicted probability p_{ij}:

$$\mathcal{L}_{rank} = -|\lambda_{ij}| \left(\overline{p}_{ij} log(p_{ij}) + \overline{p}_{ji} log(p_{ji})\right),$$

where the weight λ_{ij} corresponds to the change of ranking quality when swapping the position of d_i and d_j.

Prediction loss. To further optimize the parameters, we take additional supervised information into account so as to train the personalized language model more finely. Based on the assumption that the next query in the session expresses the real intent more accurately than previous queries, we regard the next query as the supervision to optimize the model. In other word, we are convinced that the predicted user intent based on existing queries should be close to the next query. Thus, the loss between the predicted user intent q^p and the next query q_{t+1} is:

$$\mathcal{L}_{pred} = 1 - \text{sim}(q^p, q_{t+1}),$$

where q_{t+1} refers to the average vector of each word in the next query, and sim$(\cdot)$ is cosine similarity. Note that if the current query is the last one of the session, the target query is still itself.

Finally, we minimize the ranking loss and prediction loss together to update the parameters in the model:

$$\mathcal{L} = \mathcal{L}_{rank} + \alpha \mathcal{L}_{pred}.$$

where α is a hyper-parameter to control the balance. We adopt the Adam optimizer to minimize the final loss $\mathcal{L}$.

4 EXPERIMENTAL SETUP

4.1 Dataset

We conduct our experiments on AOL search log [23] and the dataset from a commercial search engine (written as "B dataset" in the following). The basic statistics are shown in Table 1.

AOL dataset contains three months of user click data. Since this dataset only has the clicked documents, following [1], the candidate documents are selected from the top documents ranked by BM25 algorithm [26]. We also follow the method used in [1] to split log into sessions: the boundaries are identified based on the similarity between two consecutive queries. Each piece of data includes a user ID (anonymous), a session ID, a query, a document, and a click tag. Since the basis for personalized search is user historical information, we divide the whole dataset into historical data and experimental data. The former is regarded as the basic context contributing to the user's current information need, which contains the first five weeks data. The last eight weeks data is considered as experimental data, which is divided into training set, validation set and test set in a 6:1:1 ratio. Following [2, 14], we sample 50 candidate documents per query in the test set, and 5 candidates per query for training and validation. We simply use the document title to compute the relevance. To ensure the validity of the data, we remove users whose historical data or training set is empty.

B dataset contains two months of query logs in 2013, when personalization support was not applied. Different from AOL dataset, the candidate documents in this dataset are directly returned by the search engine. We use 30 minutes of inactivity as the basis for segmenting sessions following [38]. For dataset partitioning, the first six weeks is regarded as the historical data and the last two weeks is used to experiment. Since this dataset contains information on click dwell time, we consider clicks with a dwell time more than 30s or the last one in the session to be the satisfied click.

4.2 Baselines

We compare our model with adhoc search models, session search models and previous personalized search models. The original ranking and the baseline models are set as follows:

Ori. [26]. For AOL dataset, following [1], we take the ranking retrieved by BM25 algorithm as a basic baseline. For B dataset, we take its original ranking as the basic baseline directly.

KNRM [40]. It is a neural ranking model which extracts the features of interaction between query and document terms. The kernel-pooling is used to provide soft match signals for ranking.

Conv-KNRM [9]. This model is an upgrade of the KNRM model, which adds a convolutional layer for modeling n-gram soft matches. It fuses contextual information of surrounding words and learns context-aware word embeddings for matching.

BERT (Rep) [25]. It represents the query and document with the pre-trained BERT model separately, and computes the cosine similarity of them as the relevance score.

BERT (Last) [25]. This model works on the concatenated query-document sequence. It provides a "[CLS]" token at the start of the sequence and a "[SEP]" token as the joiner, and feeds them into the pre-trained BERT model. We take the last layer's "[CLS]" as the matching features and combine them linearly.

CARS [2]. It is a context-aware neural ranking model, which exploits the search activities in current session to model the user intent. Document ranking task and query suggestion task are enhanced by rich context and the whole framework is optimized via multi-task learning. We replace its GloVe word vectors with word2vec for a fair comparison with others.

P-Click [11]. This personalized search model uses the feature of click number on the same document and original position to re-rank the results via borda count. It focuses on the user's re-finding behavior and is reliable for improving results.

SLTB [4]. This method aggregates the click features, topical features, time features and position features to personalize the results via learning to rank method.

Table 2: Overall performance of all models on AOL and B dataset. The percentage reflects improvements over PSGAN. "†" indicates the model outperforms all baselines significantly with paired t-test at $p < 0.05$ level. Best results are shown in bold.

Task	Model	AOL dataset						B dataset							
		MAP		MRR		P@1		MAP		MRR		P@1		P-improve	
Adhoc Search	No context														
	Ori.	.250	-54.4%	.258	-53.9%	.148	-69.7%	.740	-9.1%	.751	-8.8%	.616	-14.1%	-	-
	KNRM	.429	-21.7%	.439	-21.6%	.271	-44.6%	.492	-39.4%	.500	-39.2%	.285	-60.3%	.066	-73.5%
	Word-based Context														
	Conv-KNRM	.474	-13.5%	.485	-13.3%	.327	-33.1%	.587	-27.7%	.598	-27,3%	.419	-41.6%	.142	-43.0%
	BERT (Rep)	.112	-79.6%	.117	-79.2%	.030	-93.9%	.201	-75.2%	.214	-74.0%	.082	-88.6%	.011	-95.5%
	BERT (Last)	.483	-11.9%	.493	-12.0%	.335	-31.5%	.573	-29.4%	.582	-29.3%	.397	-44.6%	.137	-45.0%
Session Search	Short-term Context														
	CARS	.494	-9.8%	.503	-10.2%	.352	-28.0%	.602	25.9%	.606	-26.4%	.426	-40.6%	.182	-26.9%
Personalized Search	User profile based Methods														
	P-Click	.422	-22.9%	.430	-23.3%	.379	-22.6%	.750	-7.7%	.763	-7.3%	.626	-12.8%	.061	-75.5%
	SLTB	.507	-7.5%	.519	-7.3%	.466	-4.8%	.792	-2.6%	.800	-2.8%	.690	-3.8%	.117	-53.0%
	HRNN	.542	-1.0%	.555	-1.0%	.485	-0.8%	.805	-0.9%	.819	-0.5%	.713	-0.7%	.240	-3.4%
	PSGAN	.548	-	.560	-	.489	-	.812	-	.823	-	.717	-	.249	-
	Our Context-aware Methods														
	HTPS (static)	.672†	+22.6%	.686†	+22.5%	.593†	+21.2%	.818†	+0.6%	.830†	+0.8%	.724†	+0.9%	.252†	+1.1%
	HTPS	**.709**†	+31.1%	**.723**†	+29.1%	**.627**†	+28.1%	**.822**†	+1.0%	**.832**†	+1.1%	**.729**†	+1.6%	**.255**†	+2.4%

HRNN [12]. This study uses a hierarchical recurrent neural network with query-aware attention model to build the user profile. It focuses on the sequential information hidden in query logs and dynamically adjusts the user profile based on the current query.

PSGAN [18]. It is a personalized framework based on generative adversarial network, whose aim is to extract high quality negative examples from noisy data to train the model. In our experiment, considering the cost of training, we implement the document selection based model and take the discriminator as the ranker.

The first two models have no contextual information while the next three methods utilize the word-level context to learn the word embedding. CARS takes account of the context of current session to model the query intent. The last four models are personalized approaches incorporating the influence of long-term history, but they focus more on building user profiles based on the history.

For our context-aware method, we initialize the word embedding matrix with the pre-trained vectors and fine-tune it during the training. Since the previous work [25] has revealed that BERT is not suggested to be used as a representation model, we train a word2vec [21] model to represent the query and document following [12, 18]. Specifically, our models we will experiment with are:

HTPS (**H**ierarchical **T**ransformer for **P**ersonalized **S**earch). This is the whole framework proposed in Section 3.

HTPS (static). Since the word embedding matrix of previous personalization methods [12, 18] are fixed, for fair comparison, this model keeps static word vectors without fine-tuning.

To determine the parameters of the model, we conducted multiple sets of experiments. The final parameters are selected as follows. The word embedding size is 100. The hidden size of transformer is 512. The number of heads in multi-head self-attention is 8. The number of MLP hidden units is 256. The number of kernels in matching is 11. The balance factor α is 0.5. The learning rate is $1e^{-3}$.

4.3 Evaluation Metrics

Suppose that the clicked documents for AOL dataset and the satisfied clicked documents for B dataset are relevant, and the others are irrelevant, we choose three common metrics to measure the ranking quality to evaluate the performance of different models: mean average precise(MAP), mean reciprocal rank (MRR), and precision@1 (P@1). Moreover, we form the inverse document pairs following [12, 18] to measure reliable improvements in re-ranking. The basic reason is that a relevant document may also be ignored because of a lower position [15]. To eliminate the effect of position bias, we take the percentage of improvement on inverse pairs P-improve to evaluate the results. Since the original ranking of AOL dataset is not retrieved by the search engine directly, the results on P-improve over BM25 is unreliable. So we only test this metric on B dataset, which reflects the user's click behavior in real situations.

5 RESULTS AND ANALYSIS

5.1 Overall Performance Comparison

The overall results on two datasets are reported in Table 2. It can be observed that:

(1) The comparison of user profile based personalized search methods and our context-aware methods. Our proposed methods HTPS and HTPS (fix) outperform all previous personalized search models on both datasets. Compared with the best personalized baseline model PSGAN, our models have significant improvements in all evaluation metrics with paired t-test at $p < 0.05$ level. Concretely, for AOL dataset, HTPS outperforms PSGAN by over 31.1% improvement on MAP, while the improvement percentage is 1.0% for B dataset. The reason for improvement reduction is that the B dataset has a much higher quality of original ranking than AOL dataset. It is hard to improve the results on such a baseline, so that the metric P-improve is more convincing on which our model HTPS

Table 3: Performance of ablation models on AOL dataset.

Model	MAP		MRR		P@1	
w/o. QDM	.686	-3.24%	.699	-3.32%	.600	-4.31%
w/o. PLM	.697	-1.69%	.711	-1.66%	.615	-1.91%
w/o. GT	.706	-0.42%	.720	-0.42%	.623	-0.64%
w/o. PL	.705	-0.56%	.719	-0.55%	.622	-0.80%
HTPS	**.709**	-	**.723**	-	**.627**	-

increases by 2.4% over PSGAN. These results prove that encoding user's search history as context to enhance the representation of the current query is more effective than building user profiles.

(2) The comparison of different context-aware methods. A discovery from the results of models with word-based context, short-term context and our context methods is that more contextual information is conducive to learn the context-aware word (or query) representations more accurately for document ranking. Specifically, our complete context-aware method HTPS, which aggregates the influence of long-term history, has significant improvements over CARS. An interesting phenomenon is that the model BERT (last) achieves the best results in adhoc search, whereas the performance of model BERT (Rep) is close to random. A possible reason is that BERT (Rep) discards the cross sequence interactions and the BERT model is more suitable for interactive learning.

(3) The comparison of HTPS (fix) and HTPS. The model HTPS (fix) outperforms the baseline models with the fixed word embedding matrix HRNN and PSGAN on both datasets. This illustrates that the structure of our context-aware model is effective for search results personalization. As we fine-tune the word vectors, the model HTPS further improves the quality of re-ranking. This indicates that trainable word vectors are useful in refining the representation of the current query.

(4) The comparison of different search tasks. Deep learning based personalized search methods outperform the session search and adhoc search baselines, which proves the contribution of long-term history. Specifically, we find that the improvement on the metric P@1 is more obvious than the other metrics. A possible reason is that long-term history provides strong support for the re-finding behavior. It is easy to deduce the user intent of repeated queries but is difficult for new queries that lack relevant history. Additionally, some adhoc and the session search models have better performance than the feature-based personalized methods P-Click, which shows the advantages of deep learning in semantic representation.

In summary, the experimental results prove that **encoding user's search history with context-aware representation learning, which is implemented by hierarchical transformer, is conducive to refinement of the current query's representation and search results personalization.** For more detailed analysis of the model, we conduct the following supplementary experiments: the ablation analysis, effect of short-term and long-term history, and performance on different query sets. For convenience, these experiments are performed on AOL dataset.

5.2 Ablation Analysis

To prove the effectiveness of main components in the model HTPS, we conduct ablation studies on the query disambiguation model, the personalized language model, the gated joint and the prediction loss on AOL dataset. Specifically, we remove one component at a time for performance comparison in the following.

HTPS w/o. QDM. We abandon the query disambiguation model (QDM) and take the predicted user intent from personalized language model to compute the personalized score.

HTPS w/o. PLM. We discard the personalized language model (PLM) and match the document to the refined queries from multi-stage disambiguation.

HTPS w/o. GJ. We replace the gated joint of the two models with adding their outputs directly.

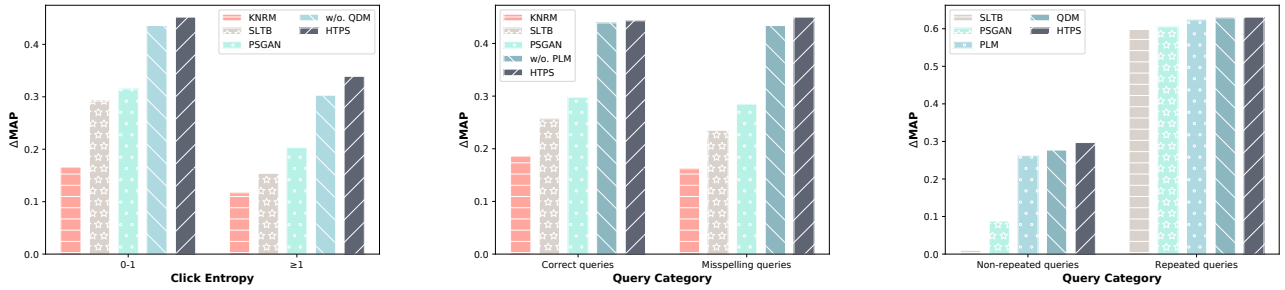
HTPS w/o. PL. We remove the prediction loss and optimize the model only depending on the ranking loss.

The ablation results are shown in Table 3. It can be seen that the results of four ablation strategies underperform the whole framework. Specifically, removing the query disambiguation model causes the most decline on MAP, which confirms the necessity and contribution of it on refining the representation of current query. The damage to the results caused by discarding the personalized language model shows that the predicted user intent contributes to the further enhancement of the query representation. Furthermore, the contributions of gated joint and prediction loss are less than the two sub-models, but still have a certain impact on the results. This proves the effectiveness of the interaction of the two sub-models and the additional supervision information on PLM. However, even if one component is removed, the gap between our model and PSGAN is still large. In addition to the effect of fine-tuning word embedding matrix, another possible reason is that user profile based personalized approaches ignore the interaction of queries and documents at the word-level. Our model applies the interaction-based matching and word-level disambiguation to capture the dependencies among terms. These results prove the necessity of each component in our model, especially the two sub-models.

5.3 Effect of Short-term and Long-term History

We previously discussed the contribution of main components in HTPS. To explore the impact of different histories, we keep one of the short-term history and long-term history at a time for experiments. Concretely, we set the HTPS-S model to test the effect of short-term history, which eliminates the long-term disambiguation stage in query disambiguation model and the low-level transformer in personalized language model. Similarly, the model that only retains the long-term history is called HTPS-L. It discards the short-term disambiguation and high-level transformer, and regards the current query as the first one of the session. Meanwhile, to delve deeper into the impact of different histories, we perform this experiment on the two sub-models QDM and PLM.

As shown in Table 4, removing either the short-term history or long-term history can considerably reduce the results. And the decline of abandoning long-term history is more obvious, which indicates that the long-term history can provide more contextual information for the moment than the short-term. Comparing the two sub-models, we find that QDM-L and PLM-L have similar declines without short-term history. But discarding the long-term history causes more damage on PLM. A possible reason is that PLM relies more on long-term history to find the personalized search patterns. Furthermore, HTPS-S, which can be seen as a session



(a) Click entropy study (b) Spelling study (c) Re-finding study

Figure 3: The results on different query sets on AOL dataset

Table 4: Effect of different histories on AOL dataset.

Model	MAP		MRR		P@1	
The whole framework						
HTPS-L	.658	-7.20%	.671	-7.19%	.564	-10.05%
HTPS-S	.628	-11.42%	.641	-11.34%	.521	-16.91%
HTPS	.709	-	.723	-	.627	-
Query disambiguation model						
QDM-L	.653	-6.31%	.669	-6.04%	.560	-8.94%
QDM-S	.625	-10.33%	.638	-10.40%	.520	-15.45%
QDM	.697	-	.712	-	.615	-
Personalized language model						
PLM-L	.641	-6.56%	.655	-6.29%	.547	-8.83%
PLM-S	.606	-11.66%	.618	-11.44%	.493	-17.83%
PLM	.686	-	.699	-	.600	-

search model, outperforms the baseline model CARS. This confirms the effectiveness of our transformer-based structure to model the user's information need.

5.4 Performance on Different Query Sets

For proving the main contribution of our model, we divide the queries in the test set into two subsets based on three different scenarios, including whether the query is ambiguous, misspelled or repeated. We test the performance of models on different query sets on AOL dataset in the following.

Ambiguous and non-ambiguous queries. We first categorize the test query sets into ambiguous and non-ambiguous queries. The former usually refers to queries that can be interpreted into multiple meanings, such as "Apple". The latter are queries with clear intents and their meanings are identical for different users. We take the click entropy [11] to measure query ambiguity and divide the queries with the threshold at 1.0. Larger click entropy means greater ambiguity and more potential for personalization. We choose baseline models KNRM, SLTB, PSGAN and our model HTPS w/o. QDM, HTPS to compare. Here we use the improvement of MAP over BM25 to show the performance.

From Figure 3(a), we observe that KNRM has a relatively balanced effect on the two sets. And personalized models improve much more on non-ambiguous queries, which is inconsistent with the conclusion in [12, 18]. The most likely reason is that the improvements is based on BM25 ranking, which can't even return satisfied results for navigational queries (lower click entropy). Compared with the best baseline model PSGAN, our models performs better on both sides, and contributes more on ambiguous queries (larger click entropy). This confirms the ability of our model to disambiguate queries. In addition, if we remove the query disambiguation model, there is a noticeable drop in performance on ambiguous queries. This is consistent with our goal of designing this sub-model.

Misspelling and correct spelling queries. Next we categorize the test query sets based on the query spelling. According to our statistics, about 27.8% of issued queries are misspelled. Interpreting the intent of these queries is hard on account of their wrong semantics. It is worthwhile to test our framework on these misspelling queries. We apply a spelling checker called enchant to divide the queries. Besides the same three baseline models as above, we choose our model HTPS w/o. PLM, HTPS here to compare.

As shown in Figure 3(b), the improvements of KNRM and SLTB on correct spelling queries are larger than misspelling queries. The deep learning-based personalized model PSGAN bridges the gap between them, but they are still unbalanced. Our complete model HTPS successfully make progress on both sets, especially on misspelling queries. This fully illustrates the ability of our model to predict user intent when facing the queries that deviate from the true information needs. It can be seen from the decline when discarding the personalized language model that this sub-model contributes more on misspelling queries.

Repeated and non-repeated queries. We finally categorize the test query sets into repeated and non-repeated queries. For the repeated queries, it is easy to infer user behaviors based on user click data on the same queries in the past. But for the non-repeated queries, there is a lack of information that can be directly referenced. To progressively observe the effects of different models, we choose the model SLTB, PSGAN and our context-aware models PLM, QDM, and HTPS for experiments.

The results shown in Figure 3(c) indicate that all of the models perform better on the repeated queries. The feature-based model SLTB has little improvement on non-repeated queries. Endowed with the benefit of deep learning, PSGAN enhances the personalized results on this part. The effect of our context-aware models are better than all baselines on both query sets, while the improvements on the non-repeated queries are more obvious. This phenomenon

indicates that our models can make use of the contextual information to infer the intent when the user submits a new query. Under the interaction of the two sub-models, our complete model HTPS achieves the best results on both query sets.

6 CONCLUSION

In this work, we propose the context-aware personalized model HTPS which encodes user's search history as contextual information to enhance the query representation. Firstly, we design a multi-stage query disambiguation model to parse the meaning of the current query. To further supplement user's information needs, a personalized language model is constructed to predict user intent based on existing queries. These two sub-models are implemented by hierarchical transformer to encode rich contextual information. After gated joint of them, we devise two loss functions to optimize the whole framework. Experimental results confirm the effectiveness of our model for personalized search.

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. (2018).
[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. *arXiv preprint arXiv:1906.02329* (2019).
[3] Paul N Bennett, Krysta Svore, and Susan T Dumais. 2010. Classification-enhanced ranking. In *Proceedings of the WWW'2010*. ACM, 111–120.
[4] 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*. ACM, 185–194.
[5] Christopher 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 the 22nd International Conference on Machine learning (ICML-05)*. 89–96.
[6] 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.
[7] Mark J. 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.
[8] Steve Cronen-Townsend and W Bruce Croft. 2002. Quantifying query ambiguity. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., 104–109.
[9] 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 eleventh ACM international conference on web search and data mining*. ACM, 126–134.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[11] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW'2007*. ACM, 581–590.
[12] 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 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*.
[13] Morgan Harvey, Fabio Crestani, and Mark J Carman. 2013. Building user profiles from topic models for personalised search. In *CIKM'2013*. ACM, 2309–2314.
[14] Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. 2018. Improving Entity Recommendation with Search Log and Multi-Task Learning.. In *IJCAI*. 4107–4114.
[15] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR'2005*. 154–161.
[16] Lin Li, Zhenglu Yang, Botao Wang, and Masaru Kitsuregawa. 2007. Dynamic adaptation strategies for long-term and short-term user profile to personalize search. In *Advances in Data and Web Management*. Springer, 228–240.
[17] Xiujun Li, Chenlei Guo, Wei Chu, Ye-Yi Wang, and Jude Shavlik. 2014. Deep learning powered in-session contextual ranking using clickthrough data. In *NIPS'2014*.
[18] 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 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*.
[19] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1101–1104.
[20] Ryan McDonald, Georgios-Ioannis Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. *arXiv preprint arXiv:1809.01682* (2018).
[21] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* (2013).
[22] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
[23] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search.. In *InfoScale*, Vol. 152. 1.
[24] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).
[25] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).
[26] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
[27] Ahu Sieg, Bamshad Mobasher, and Robin Burke. 2007. Web search personalization with ontological user profiles. In *CIKM'2007*. ACM, 525–534.
[28] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. 1999. Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, Vol. 33. ACM, 6–12.
[29] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep ranknet for personalized search. In *WSDM'2014*. ACM, 83–92.
[30] David Sontag, Kevyn Collins-Thompson, Paul N Bennett, Ryen W White, Susan Dumais, and Bodo Billerbeck. 2012. Probabilistic models for personalizing web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 433–442.
[31] Jaime Teevan, Daniel J Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *WSDM'2011*. ACM, 85–94.
[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[33] Maksims Volkovs. 2015. Context models for web search personalization. *arXiv preprint arXiv:1502.00527* (2015).
[34] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *ECIR'2017*. Springer, 598–604.
[35] Thanh Vu, Dawei Song, Alistair Willis, Son Ngoc Tran, and Jingfei Li. 2014. Improving search personalisation with dynamic group formation. In *SIGIR'2014*. 951–954.
[36] Thanh Vu, Alistair Willis, Son N Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *ECIR'2015*. Springer, 605–616.
[37] Ryen W White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *WWW'2013*. ACM, 1411–1420.
[38] Ryen W White and Steven M Drucker. 2007. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 21–30.
[39] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. 2008. *Ranking, boosting, and model adaptation. Tecnical Report*. Technical Report. MSR-TR-2008-109.
[40] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*. ACM, 55–64.
[41] Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972* (2019).
[42] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3481–3487.