# Clarifying Ambiguous Keywords with Personal Word Embeddings for Personalized Search

## ABSTRACT

Personalized search is a task to tailor the general document ranking list based on user interests and return each user a personalized result list which can better satisfy the user's information need. Many personalized search models have been proposed, and demonstrated their capability to improve search quality. The general idea of most approaches is to build a user interest profile according to the user's search history, and then re-rank the documents on the basis of the matching scores between the created user profile and the candidate documents. In this paper, we propose to solve the problem of personalized search in an alternative way. We know that there are many ambiguous words in natural language such as 'Apple', and people with different knowledge background and interests have different understandings of these words. Therefore, for different users, such a word should own different semantic representations. Motivated by this idea, we design a personalized search model based on personal word embeddings, refereed as PEPS. Specifically, we train personal word embeddings for each user in which the representation of each word is mainly decided by the personal data of the user. Then, we obtain the personalized word and contextual representations of the query and documents with the attention function. Finally, we use a matching model to calculate the matching score between the personalized query and document representations. Experiments on two datasets, including the publicly available AOL dataset and a commercial query log, verify that our model can significantly improve the state-of-the-art personalized search models.

## KEYWORDS

personalized search, personal word embedding, user interest

## 1 INTRODUCTION

Search engine is one of the major approaches for us to obtain information from the Web in our daily life. Given a query, it returns a ranked document list in which documents are ordered by their relevance to the query. Obviously, it is not an optimal solution to return the same search results to various users for the same query, because there are many ambiguous words in natural language and different users may have different query intents when they enter these keywords. Let us take the query 'Apple' as an example. An IT engineer may use this query to search for information about the 'Apple' company or products, while a fruit farmer tends to seek information related to the 'Apple' fruit using the same query. From this example, we can find that the IT engineer and the fruit farmer have different understandings of the word 'Apple' due to their own knowledge background and interests. Returning more specific ranking results to each user based on the user's interests can improve result quality and user satisfaction, and this is the target of personalized search.

Many models have been proposed for search results personalization. Traditional personalized search models [5, 6, 10, 11, 13, 16, 19, 37–39] mainly depend on click-based and topic-based features extracted from the search history to analyze user interests. With the emergence of deep learning, new personalized models [17, 24] have achieved better personalization by learning user interest profiles based on neural networks. The common idea of most existing personalized search methods is building a user interest profile with the search history at first and then tailoring the general document list on account of the matching scores between the created user profile and candidate documents. In this paper, **we attempt to solve the problem of search results personalization from an alternative perspective of clarifying the ambiguous keywords with personal word embeddings**.

As shown in the previous example, there are lots of ambiguous words in queries and different users have different understandings of such words' meanings due to their own knowledge background and interests. To clarify the specific personalized meaning of such ambiguous words, we claim that the same word for different users should be viewed as different words and own different semantic representations. Therefore, we propose a personalized model which trains personal word embddings enhanced from the global word embeddings for each user with her own search history as the training data. This model has several advantages in solving the problem of personalized search. **First**, the embedding of each word trained on the user's search log is personalized which mainly contains the meaning that the user is interested in. **Secondly**, the user's personalized query intent can be clarified by the contextual vector of the query represented with the personal word embeddings. Thus, the documents which meet the user's query intent can be matched with higher scores. **Thirdly**, we can directly re-rank the documents based on the relevance scores between the personalized representations of the query and documents. The user interests reflected in the search history have been contained in the personal word embeddings after training the model with the query logs. Hence it is not necessary for us to analyze the user's search history and build the user interest profile every time we re-rank the documents for a new query issued by the user, saving a large amount of time for maintaining the user profile.

Specifically, we design a personalized search model PEPS (**P**ersonal **E**mbedding based **P**ersonalized **S**earch model), which consists of four modules. To begin with, we construct a personalized word embedding layer, including embedding matrices for each individual

*Woodstock '18, June 03–05, 2018, Woodstock, NY*
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

users, to get personalized representations at the word level. Then, personalized contextual representations of the query and documents are calculated through a multi-head self-attention layer to fuse the contextual information. With personalized word and contextual representations ready, we compute the matching score for document ranking with the KNRM component. Finally, we utilize two tasks, personalized document ranking and query reformulation, to jointly train the personal word embeddings and ranking model in an end-to-end way. Before model training, we initialize the personal word embeddings with word2vec [25] trained on the logs of similar users or the global word2vec model. And we devise online update approaches to track the new user interests along with the search process. To compare our PEPS model with state-of-the-art baselines, we conduct experiments on the public AOL dataset and a query log from a commercial search engine. Experimental results show that our method can yield significant improvements on ranking results and performance over existing model.

To conclude, our main contributions are three-fold: (1) We attempt to solve the problem of search results personalization from an alternative perspective of clarifying the ambiguous keywords with personalized word representations, without building user interest profiles. (2) Based on this idea, we propose a personalization model to train separate personal word embeddings for each user, and compute ranking scores by matching the personalized representations of the query and documents. (3) We design three approaches to update the personal word embeddings to capture the changes of user interests reflected in the newly issued queries.

The rest of the paper is organized as follows. Related works are introduced in Section 2. In Section 3, we elaborate each component of our proposed model. We describe the experimental settings in Section 4, present and analyze the experimental results in Section 5. Finally, the whole paper is concluded in Section 6.

## 2 RELATED WORK

Search results personalization has been proved to effectively improve the quality of search results, especially for those ambiguous queries [16, 17, 33, 34], and there is a wealth of related studies.

*Traditional Personalized Search Model.* Traditional personalized models rely on some heuristic rules to analyze user interests. Motivated by the user's refinding behaviors, Dou et al. [16] proposed P-Click model which evaluates the relevance by counting how many times the documents have been clicked by the same user under the same query. Many models [11, 19, 29, 37, 38, 41] applied a topic model such as Latent Dirichlet Allocation (LDA) [7, 41] to extract topic-based features from the clicked documents and issued queries, then built user interest profiles in the topic space. Some other studies [6, 36, 39] realized personalization through feature engineering. They extracted many features from the current query and the user's search history, including the original rank position, click-based features and so on. Then, learning to rank algorithms [8, 9] are used to combine these features to train a ranking model. In addition to the features related to queries and clicks, location information and the user's reading level [5, 13] were also used. Traditional personalized models have achieved great progress, but most of them only focus on using some specific features to describe user interests, thus ignoring other information that is the same valuable.

*Deep Learning based Personalized Search.* As deep learning becomes popular, many personalized search models based on learning have been studied, and the problem that the representation ability of traditional manual-designed features is limited has been relieved gradually. These learning-based models mainly follow two kinds of approaches. One is the adaptation framework [30], which adapts the general ranking model to a personalized model by training with a few queries from that user. The other more common method is to learn an explicit representation of the user interest profile from the search history. Ge et al. [17] designed a hierarchical RNN model (HRNN) to learn both the user's long-term interests and short-term interests, obtaining a comprehensive user profile. And Lu et al. [24] devised PSGAN which applied the generative adversarial network (GAN) [18] to enhance the training data and promote the learning of user profiles. These models tailor the original document list based on user interest profiles and achieve state-of-the-art performance. In this paper, we propose a novel personalized model from a different perspective, without building user interest profiles.

*Word Embedding for Personalization.* Recent years, there are a few models [3, 20, 23, 26, 28] attempting to apply word embeddings for personalization. Typically, Samarawickrama et al. [28] first trained personalized neural language model on the user's history to create a synonym table for the user, and then re-rank the documents based on cosine similarity between the query's synonyms and documents. Amer et al. [3] applied word embeddings to find similar words for query expansion, and computed relevance scores between the expanded query and documents with Language Model. These search models use word embeddings to find some synonyms to indicate user interests and help document re-ranking, while our model is different which directly trains personal word embeddings containing user interests and computes relevance score by matching the personalized query and document representations. These existing models trained word vectors by unsupervised language model, without the user's click labels, which are the most credible indication of user preferences. In addition, their embeddings trained with a individual user's search log are also unreliable.

Our PEPS is a novel end-to-end personalized model, which trains personal word vectors enhanced from the global word embeddings with the user's click information in a supervised way to embed the user interests into the personal word embeddings, solving the problems faced by the models above.

## 3 PEPS - A PERSONAL WORD EMBEDDING BASED PERSONALIZED SEARCH MODEL

As we stated in Section 1, most existing personalized search models either extract features related to the user interests or learn a user interest profile from the search history to achieve personalization. Differently, we propose a personalized search model from the perspective of personalized word representations to tackle this problem in an alternative way. Through training personal word embeddings for each individual user, we obtain word representations that mainly contain the meanings the user already knows or she is interested in. Using such personal word embeddings to represent the query and documents, the user's real query intent can be clarified and personalized search results can be improved.

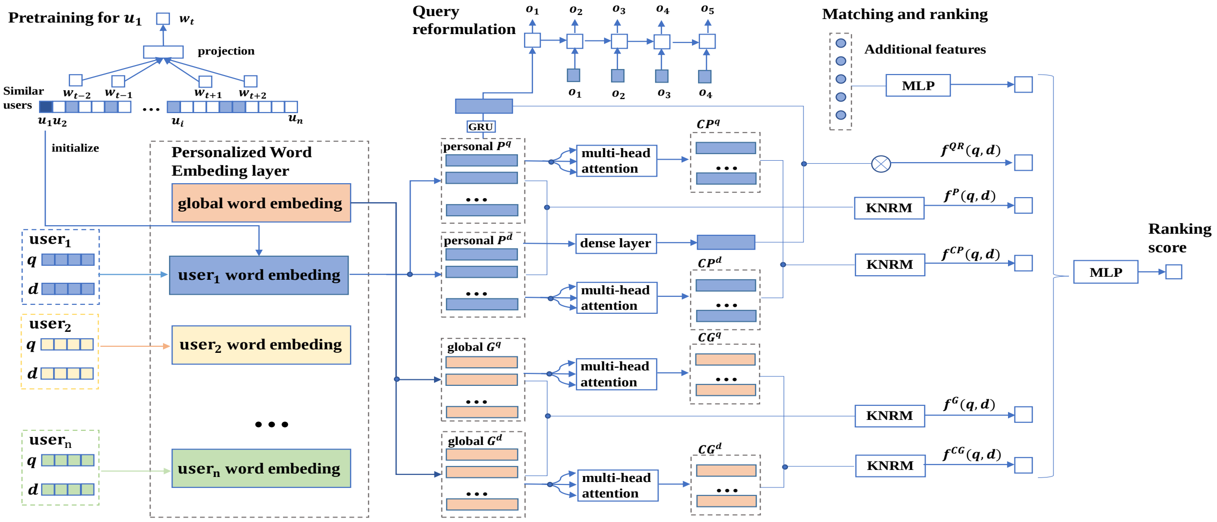

**Figure 1: Structure of the PEPS. Each user $u_i$ owns a personal word embedding matrix pretrained on the query logs of similar users. Given a query issued by $u_i$ and a document, the embedding layer maps their words to global and personal word representations, which are fed into the multi-head attention layer to obtain contextual representations. A GRU encodes the word representations to get query intent vectors used for query reformulation based on Seq2Seq model. Finally, we compute the document relevance score with neural matching component KNRM and train the model through joint learning.**

To begin with, we formulate the problem with notations. Suppose that there are a lot of individual users, represented as $u_1, u_2, \ldots$, and each user $u_i$ has her own search history $H_i^T$ at the current time $T$. The search history $H_i^T$ includes a sequence of queries $\{q_{i,1}, q_{i,2}, \ldots, q_{i,N_i^T}\}$ issued by the user $u_i$, the clicked documents set $D_{i,j}^+$ and unclicked documents set $D_{i,j}^-$ under each query $q_{i,j}$. $N_i^T$ is the total number of queries in $H_i^T$. Currently, the user $u_i$ enters a query $q$, and the underlying non-personalized search engine returns a candidate document list $D = \{d_1, d_2, \ldots\}$. Personalized search models are required to tailor the candidate document list taking user interests into account, and give higher priority to the documents that match the user's query intent and interests.

Our proposed personalized search model PEPS mainly includes three stages: the pre-training stage of the personal word embeddings, the supervised training stage of the model and the online update stage. Next, we first describe the pre-training stage, and then introduce each component of our personalization model. In the final, we present three approaches for online update in real-word situations and make some discussions about our model.

## 3.1 Pre-training

According to existing works [6, 16, 17], user interests are mainly reflected in the historical issued queries and clicked documents under each query. Therefore, the most direct approach to obtain the personalized word representations containing user interests is to train word embeddings on the personal corpus which consists of the issued queries and clicked documents [3, 28]. We implement this approach as one of our baselines, introduced as PPWE in Section 4.2. But we know that training reliable word vectors from scratch usually relies on large corpus, and the individual query log is not enough. Here, to initialize the personal word embeddings for our personalization model, we propose two methods taking both user interests and the amount of training data into consideration.

(1) We train a global word2vec model [25] on the whole query logs, and initialize the personal word embeddings for each user with this global model.

(2) Considering the user interests, we adapt the global word2vec model to a personalized model with the query logs of that user and other top $k$ users with similar interests. Then, the personal word embeddings are initialized with the adapted word2vec model. We refer to the user-based collaborative filtering algorithm [40] to find users with similar interests. We create a user similarity matrix $W$ and the interest similarity between two users $u_i$ and $u_j$ represented as $W_{ij}$ is calculated as:

$$W_{ij} = \frac{\sum_{d \in N(u_i) \cap N(u_j)} \frac{1}{\log(1+|N(d)|)}}{\sqrt{|N(u_i)||N(u_j)|}}. \tag{1}$$

$N(u_i)$ and $N(u_j)$ represent the clicked document sets of user $u_i$ and $u_j$, and $N(d)$ is the set of users who clicked the document $d$.

## 3.2 Personalization Model

After the pretraining stage, we obtain rough but not exactly accurate personal word embeddings for each user. Because word2vec [25] is an unsupervised model, it can capture the co-occurrence or semantic relationships between words but is hard to learn user interests reflected in the user's click behaviors. Therefore, we further design a supervised personalization model to finetune the personal word embeddings with the user's click information, obtaining personal word vectors which really contain user interests.

The whole architecture of our model is illustrated in Figure 1, and we divide it into four parts. In the first part, we set a personalized embedding layer with a global word embedding matrix and personal word embeddings for each individual user. In the second part, we get representations of the query and document from different granularities and perspectives. Thirdly, we use the neural matching component KNRM [42] to compute the match score between the query and document, and apply the pairwise LambdaRank [8] to train the ranking model. Finally, we design a query reformulation module to promote the learning of personal word embeddings. In the following, we will describe the details.

*3.2.1 Personalized Word Embedding Layer.* To obtain personalized word representations, we design a word embedding layer in our model, in which we keep a personal word embedding matrix for each user and a global embedding matrix, shown as the left module in Figure 1. We use two signs to identify a word, i.e. the word and the corresponding user id, so that the same word of different users are identified as different words. For example, the word 'Apple' in the word embedding matrix of the user $u_i$ should be represented as 'Apple + $u_i$', while 'Apple + $u_j$' is for the user $u_j$. In this layer, we fine-tune the personal word embeddings with only the corresponding user's click data. Thus, **the well-trained embedding of a specific word is not a general representation of various meanings of this word in the overall logs, but mainly the personalized meaning that the user is interested in.**

We also have to decide the personal vocabulary for each user. There is a global vocabulary on the whole query logs, but copying a global vocabulary for each user as the personal vocabulary yields several drawbacks: (1) Most words in the global vocabulary don't appear frequently in a user's query log, so it is unnecessary to maintain a complete global vocabulary for each user, which will take up a large amount of memory space. And (2) some words are not obviously ambiguous, thus we argue that it is better to train the embedding of these words on the whole logs. Based on these considerations, we keep a public global word embedding matrix for all users and train it with the whole query logs. As for the personal vocabulary, we filter the words according to several rules:

- words that are not stop words
- words that occur in the user's query log more than $c$ times
- words with word entropy no less than a threshold *ent*

In this paper, we define the word entropy of a word $w$ as the average click entropy [16] of all queries containing the word, computed as:

$$\text{WordEntropy}(w) = \frac{\sum_{q \in Q(w)} \text{ClickEntropy}(q)}{|Q(w)|}, \tag{2}$$

$$\text{ClickEntropy}(q) = \sum_{d \in D(q)} -P(d|q) \log_2 P(d|q). \tag{3}$$

Here, $Q(w)$ represents the set of queries that contain the word $w$, and $D(q)$ is the collection of documents clicked under the query $q$. $P(d|q)$ is the probability of the clicks on document $d$ among all clicks of the query $q$, calculated as $P(d|q) = \frac{|\text{Clicks}(q,d)|}{|\text{Clicks}(q,\cdot)|}$.

With the selection of personal vocabulary, we can effectively control the space of storage used by personal word embeddings and the computation cost for updating the embeddings.

*3.2.2 Query and Document Representation.* With the well-designed personalized word embedding layer, we are able to map the query $q = \{w_1^q, w_2^q, \ldots\}$ and document $d = \{w_1^d, w_2^d, \ldots\}$ into a high-dimensional vector space and obtain their text representations. In our model, the text representations are composed of four parts.

(1) **Personalized word representation**: We obtain this part of representation by passing the query and document through the corresponding user's personal word embedding matrix, getting $P^q \in R^{dim \times |q|}$ for the query and $P^d \in R^{dim \times |d|}$ for the document. The word vector mainly contains the meanings that the user knows or is interested in, achieving personalization at the word level.

(2) **Personalized contextual representation**: To model the interactions between contexts and obtain personalized representation at the query level to further clarify the personalized query intent, we use a multi-head self-attention layer [35] on the top of the personalized word representations outputted by the embedding layer, obtaining two matrices $CP^q \in R^{dim \times |q|}$ and $CP^d \in R^{dim \times |d|}$. As for the calculation of the personalized query contextual representation $CP^q$, the input is the query word vectors $P^q$. We first process it with different linear functions to $d_q$, $d_k$ and $d_v$ dimensions respectively. Then, we conduct the attention function on the processed results of different heads in parallel, yielding several $d_v$ dimensional output values. These outputs are concatenated together and processed again with a dense layer to get the final attended result $CP^q$. The concrete computation formulas are listed as follows:

$$CP^q = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^A, \tag{4}$$

$$\text{head}_i = \text{softmax}(\frac{P^q W_i^Q (P^q W_i^K)^T}{\sqrt{d_k}})(P^q W_i^V), \tag{5}$$

where $W_i^Q, W_i^K, W_i^V$ and $W^A$ are parameters of linear functions. In the same way, we can obtain the personalized contextual representation $CP^d$ for the document. $CP^q$ and $CP^d$ fuse the contextual information to further clarify the ambiguous words.

(3) **Global word representation**: In actual situations, every user's interests are variable and knowledge is growing. Like the IT engineer in our previous example, in most cases, he issues the query 'Apple' to seek for the Apple company or products. However, it is also inevitable that he would sometimes use other meanings of 'Apple' to search for information that has never been searched before, such as the apple fruit. Therefore, in addition to the personalized representations, we also pay attention to the representations in the global vector space. We get global word representations $G^q \in R^{dim \times |q|}$ for the query and $G^d \in R^{dim \times |d|}$ for the document from the global word embedding matrix.

(4) **Global contextual representation**: Similar to the personalized contextual representation, we obtain global contextual vectors $CG^q$ and $CG^d$ with multi-head self-attention.

With the four parts, we obtain comprehensive query and document representations of different granularity and aspects. They are helpful for matching the query and document more accurately.

*3.2.3 Query-Document Matching and Ranking.* With the personalized and global representations of the query and documents, we are able to compute the personalized matching scores to re-rank the original document list. In this paper, we adopt the neural matching model KNRM [42]. For the personalized word representations, we first construct a similarity matrix $S^P$ where $S_{ij}^P$ is the embedding similarity between the query word $w_i^q$ and the document word $w_j^d$ calculated by cosine similarity. Then, many RBF kernels are used on the personalized word similarity matrix $S^P$ to extract multi-level soft-match features $\phi(S^P)$ between the query and document.

$$\phi(S^P) = \sum_{i=1}^{|q|} \log(\vec{K}(S_i^P)), \tag{6}$$

$$\vec{K}(S_i^P) = \{K_1(S_i^P), \ldots, K_K(S_i^P)\}, \tag{7}$$

$$K_k(S_i^P) = \sum_j \exp(-\frac{(S_{ij}^P - \mu_k)^2}{2\sigma_k^2}). \tag{8}$$

Here $|q|$ is the query length, and $K$ is the number of RBF kernels. $\mu_k$ and $\sigma_k$ are the center and variance of the $k^{th}$ kernel.

After obtaining a series of query-document ranking features $\phi(S^P)$, we use a multi-layer perceptron (MLP) with the tanh(·) activation function to combine all these features and compute the matching score $f^P(q,d)$ between the personalized word representations of the query and document, i.e.:

$$f^P(q,d) = \tanh(W_P^T \phi(S^P) + b^P) \tag{9}$$

Same as the calculation process above, we further use three KNRM components with different parameters to compute the matching scores for other query and document representations in the last section, getting three scores $f^{CP}(q,d)$, $f^G(q,d)$ and $f^{CG}(q,d)$.

In addition to the interactive match scores between the query and document, we also calculate a match score $f^{QR}(q,d)$ of the query hidden state in the query reformulation module and the document by cosine similarity. And we follow [17, 24] to incorporate some click-based features and relevance features to help ranking, totally 98 dimensions, which are proved to be effective in [6]. We input these features into a MLP with the tanh(·) activation function to calculate a relevance score. Finally, all these six scores are combined with a MLP to get the personalized score for the document.

We apply a pairwise LTR algorithm LambdaRank [8] to train our ranking model. First, we create training document pairs on the whole logs, with the clicked documents as positive samples and the skipped documents as negative ones. $\lambda$ for each document pair is the change value of the metric MAP when swapping the positions of the two documents. And the final loss function is the dot product of the $\lambda$ and cross entropy between the real probability distribution of the relative relationship and the predicted probabilities. We have:

$$loss = (-\hat{p}_{(i>j)} \log(p_{(i>j)}) - \hat{p}_{(i<j)} \log(p_{(i<j)}))|\lambda_{ij}|. \tag{10}$$

$\hat{p}_{(i>j)}$ is the true probability that $d_i$ is more relevant than $d_j$, and $p_{(i>j)}$ is the predicted probability computed as the score difference $(\text{score}(d_i) - \text{score}(d_j))$ normalized by a logistic function.

*3.2.4 Query Reformulation.* In most situations, it is difficult for the users to express their query intents with accurate queries. In our model, we have the personalized query representations so that we are able to infer the user's real query intent and reformulate the query to promote the ranking, which can also help the learning of personal word embeddings in return. Based on this motivation, we design a query reformulation module and construct a multi-task framework to jointly train it with the personalized ranking model. Due to lack of manually labeled reformulated queries which express the user's true query intents, we follow [21, 31] to use the next query in the same session, which is thought to express the user intent more accurately than the current query, as the learning target of this task. Referring to existing query generation models [21, 31], we implement the query reformulation task with the sequence to sequence structure [32]. The input is the personalized word representations of the query $P^q$ and a GRU is used as the encoder to compute the hidden state of each step $h_t = GRU(P_t^q, h_{t-1})$. The last hidden state $h_{|q|}$ of the encoder is regarded as the representation of the user's real query intent, and is used to calculate a matching score $f^{QR}(q,d)$ with the document by cosine similarity as a part of the final relevance score for ranking. In the decoder, a GRU [12] with the attention mechanism [4] is applied, and each token $o_t$ in the target sequence is predicted based on the current hidden state $s_t$, the previous decoded tokens $\{o_1, \ldots, o_{t-1}\}$ and the context vector $c_t$ computed by attention function, i.e:

$$p(o_t|\{o_1, \ldots, o_{t-1}\}, h_{|q|}) = \text{softmax}(W[s_t, c_t] + b), \tag{11}$$

where $s_t = GRU(s_{t-1}, o_{t-1}, c_t)$, $c_t = \sum_{j=1}^{|q|} \alpha_{tj} h_j$ and the weight $\alpha_{tj} = \frac{\exp(a(s_{t-1}, h_j))}{\sum_{k=1}^{|q|} \exp(a(s_{t-1}, h_k))}$. The function $a$ can be a simple dot product or a trainable MLP. Then, the probability of the target sequence $p(o)$ is defined as the joint probability of all the tokens.

$$p(o) = \prod_{t=1}^{T} p(o_t|\{o_1, \ldots, o_{t-1}\}, h_{|q|}). \tag{12}$$

We train the query reformulation module by minimizing the negative log likelihood of the target sequence. And the whole multi-task framework is optimized by minimizing the sum of the negative log likelihood and the pairwise loss.

## 3.3 Online Update

The PEPS model trained offline on the query logs has contained the user interests reflected in the search history. In real-word application scenario, users will continuously issue new queries which may show new user interests. To ensure that our personal word embeddings contain the latest user interests, we should finetune the personal word embeddings according the new issued queries along with the search process, keeping the ranking model fixed. In this paper, we design three different approaches for adjustment.

**Update by stage**: In the first step, we train a model offline with all users' search history before the current moment. In the second step, we set a fixed duration of a stage. During this stage, we collect all the click behaviors but don't change the word embeddings. Thirdly, at the end of each stage, we finetune the personal word embeddings with the collected data, keeping the other parameters of the model static. Lastly, we repeat the second and the third steps to track the latest search process and user interests.

**Update by session**: In information retrieval, we usually take sessions as search activities with independent query intents, which may reflect complete user interests. Therefore, based on the update process described above, we propose to adjust the word embeddings at a session interval.

**Update by query**: Many personalized models [6, 17] divide the user interests into long-term and short-term user interests, where the short-term interests are defined as interests in a session. The method which updates the word embeddings by sessions can't capture the impact of the short-term user interests on the subsequent queries in the same session. Thus, we also design an approach to update the word embeddings in units of queries to capture more fine-grained interests.

All the three methods are applied to finetune the personal word embeddings with the latest queries issued by users online on the basis of the offline well-trained model. But it may be difficult for the model to achieve the global optimal state by such methods of incremental finetuning, and a long-term adjustment may make the model perform worse. With this consideration, we suggest to adjust the model by the update-by-query approach in a short time, and then a large batch of new training samples can be added to the original dataset to retrain an optimal model offline, achieving a balance between the effectiveness and efficiency.

## 3.4 Discussion about PEPS

Here, we first discuss the performance and feasibility of PEPS. Although we set a personal word embedding matrix for each individual user in the model, we have stated in Section 3.2.1 that we filter the words in personal vocabulary strictly and merely some necessary ambiguous words should be maintained, which will not take up too much space and computing resources. If there are numerous users, we can set a max number of users on a single model, and distribute all users on multiple models. In addition, through embedding user interests into the personal word representations, our model is not required to keep the search history in memory and process the history to build the user interest profile when re-ranking the documents for the current query, improving the performance and saving memory space. Thus, we can conclude that our proposed model provides several advantages for personalized search: Firstly, the personal word embeddings trained on the user's query log contain the user interests, so that the ambiguous keywords and personalized query intent can be clarified by representing the query with the personal embeddings. Secondly, PEPS model improves the efficiency of personalized search significantly, without obviously increasing pressure on the space occupation.

# 4 EXPERIMENTAL SETTINGS

## 4.1 Dataset and Evaluation Metrics

We evaluate our model on two non-personalized search logs, and the statistics of the processed datasets are listed in Table 1

**AOL Dataset**: This is a publicly available query log dataset collected from $1^{st}$ March 2006 to $31^{st}$ May 2006. Each piece of data contains a user anonymous id, a query text, the time when the query was issued, a clicked url and the rank position of the url. Following [1, 2], we segment the user query sequences into sessions with boundaries decides by the similarity between two consecutive queries. To ensure that every user has enough search history for building user profile, we set the first five weeks log as the history, and the remaining data are used for model training, validating and testing with the proportion 6:1:1. AOL dataset only records clicked documents which are regarded as relevant documents under each query without unclicked documents, so we refer to [1, 2] to find irrelevant documents and construct original result list with the BM25 algorithm [27]. Following [1, 2], we construct 5 candidate documents for each query in the training and validation set, while 50 candidates for each testing query, and each document corresponds to a title. After process, the dataset includes 110,869 users. We count the number of queries issued by each user and find that most users only have a small amount of search data. Considering that we need enough individual query log to finetune the corresponding personal word embeddings, we sample about 30,000 users with the most training data and set personal word embeddings for each of them, global word embeddings for the other users.

**Table 1: Statistics of the datasets.**

| Dataset | AOL Dataset | | | Commercial Dataset | | |
|---|---|---|---|---|---|---|
| | Train | Valid | Test | Train | Valid | Test |
| #session | 187,615 | 26,386 | 23,040 | 71,731 | 13,919 | 12,208 |
| #query | 814,129 | 65,654 | 59,082 | 188,267 | 37,951 | 41,261 |
| avg query len | 2.845 | 2.832 | 2.895 | 3.208 | 3.263 | 3.281 |
| avg #click | 1.249 | 1.118 | 1.115 | 1.194 | 1.182 | 1.202 |

**Commercial Dataset**: This is a large-scale query log collected from a non-personalized commercial search engine between $1^{st}$ Jan. 2013 and $28^{th}$ Feb. 2013. Each query record contains a user id, a query string, query issued time, the top 20 retrieved urls, click labels and their dwelling time. Each url corresponds to a document with text body. Following [17], we view clicked documents with longer than 30 seconds of dwelling time as relevant documents. With 30 minutes of user inactivity as the boundary [6], we segment the search process into sessions. Then, we divide the logs in the first six weeks as the historical data and the last two weeks as the experimental data which is further split into the training set, validate set and testing set by sessions with 4:1:1 ratio. There are totally 5,998 users in the dataset and we select 4,000 users with the most training data to build personal word embeddings.

**Evaluation metrics** We apply the most widely used ranking metrics MAP, MRR and P@1 to evaluate our model. Considering the fact that users' recorded click actions are inevitably influenced by the original order and some documents are not clicked may not due to their irrelevance but their low rankings, we use a more credible metric P-Improve [24] based on reliable relevance preferences in this paper. Following [14, 22, 24], we construct inverse document pairs viewing only the documents skipped above the click and the non-clicked next document as irrelevant, and compute P-Improve as the ratio of the correctly ranked inverse pairs. We only use the P-Improve metric on the commercial dataset whose recorded document list were actually presented to users. The candidate list of AOL dataset is constructed by us with BM25 algorithm which are not the list shown to users, hence the P-Improve value calculated on the AOL dataset is unreliable. Because of this, we do not use the P-Improve metric for the AOL dataset.

## 4.2 Baselines

In addition to the original ranking (on the AOL dataset, it is generated by BM25. On the commercial dataset, it is returned by the search engine), we select several state-of-the-art adhoc ranking models and personalization models as baselines, listed as follows:

(1) **KNRM** & **Conv-KNRM**: KNRM [42] is a kernel based neural ranking model for ad-hoc search. It conducts a kernel-pooling

Table 2: **Overall performances of models. Relative performances compared with PSGAN are in percentages."†" indicates significant improvements over all baselines with paired t-test at $p < 0.05$ level, and ‡ for t-test at $p < 0.01$ level. The best results are shown in bold. PEPS(fix) means the personalized word embedding layer is fixed during training.**

| Model | AOL Dataset | | | | | | Commercial Dataset | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | | MRR | | P@1 | | MAP | | MRR | | P@1 | | P-Imp. | |
| Adhoc search model | | | | | | | | | | | | | | |
| Ori. | .2504 | -54.3% | .2596 | -53.6% | .1534 | -68.6% | .7399 | -9.1% | .7506 | -8.8% | .6162 | -14.1% | - | - |
| KNRM | .4291 | -21.7% | .4391 | -21.6% | .2704 | -44.7% | .4916 | -39.6% | .5001 | -39.3% | .2849 | -60.3% | .0655 | -73.7% |
| ConvK | .4738 | -13.5% | .4849 | -13.4% | .3266 | -33.2% | .5872 | -27.8% | .5977 | -27.4% | .4188 | -41.6% | .1422 | -42.9% |
| User profile based personalized search model | | | | | | | | | | | | | | |
| PClick | .4224 | -22.9% | .4298 | -23.3% | .3788 | -22.6% | .7509 | -7.7% | .7634 | -7.3% | .6260 | -12.7% | .0611 | -75.5% |
| SLTB | .5072 | -7.5% | .5194 | -7.3% | .4657 | -4.8% | .7921 | -2.6% | .7998 | -2.9% | .6901 | -3.8% | .1177 | -52.7% |
| HRNN | .5423 | -1.0% | .5545 | -1.0% | .4854 | -0.8% | .8065 | -0.9% | .8191 | -0.5% | .7127 | -0.7 % | .2404 | -3.4% |
| PSGAN | .5480 | – | .5601 | – | .4892 | – | .8135 | – | .8234 | – | .7174 | – | .2489 | – |
| Embedding based personalized search model | | | | | | | | | | | | | | |
| PWEBA | .4284 | -21.8% | .4368 | -22.0% | .2687 | -45.1% | .7415 | -8.9% | .7529 | -8.6% | .6201 | -13.6% | .0433 | -82.6% |
| PPWE | .6542$^{\ddagger}$ | 19.4% | .6668$^{\ddagger}$ | 19.1% | .5613$^{\ddagger}$ | 14.7% | .8138 | 0.1% | .8249 | 0.2% | .7187 | 0.2% | .2338 | -6.1% |
| PEPS(fix) | .6971$^{\ddagger}$ | 27.2% | .7107$^{\ddagger}$ | 26.9% | .6153$^{\ddagger}$ | 25.8% | .8209$^{\dagger}$ | 0.9% | .8310$^{\dagger}$ | 0.9% | .7232$^{\dagger}$ | 0.8% | 0.2516 | 1.1% |
| PEPS | **.7174**$^{\ddagger}$ | 30.9% | **.7309**$^{\ddagger}$ | 30.5% | **.6338**$^{\ddagger}$ | 29.6% | **.8221**$^{\dagger}$ | 1.1% | **.8321**$^{\dagger}$ | 1.1% | **.7251**$^{\dagger}$ | 1.1% | **.2545**$^{\dagger}$ | 2.3% |

technique on the word similarity matrix to extract multi-level soft match features, which are combined with a pairwise LTR algorithm to get the ranking score. Conv-KNRM [15] was proposed on the basis of KNRM to model n-gram soft matches with CNN.

(2) **P-Click**: Dou et al. [16] proposed P-Click to re-rank documents based on the number of clicks made by the same user under the same query in history, satisfying the user's refinding behaviors.

(3) **SLTB**: It [6] extracts 102 features from the user's search log, including click-based features, topic-based features and so on, and combines all the features with the LTR algorithm LambdaMart [9] to generate the personalized ranking list.

(4) **HRNN**: This model [17] dynamically builds both short and long-term user interest profiles with a hierarchical RNN and query-aware attention mechanism. Documents are re-ranked based on the similarities with the user profiles and the additional SLTB features.

(5) **PSGAN**: It [24] is a personalized framework which applies GAN to generate queries that match the user's query intent better and select document pairs more valuable for learning user interests. In this paper, we take the variant PSGAN-D as our baseline.

(6) **PPWE**: This is a pipeline personalized word embedding based model we implement as a baseline. To re-rank the documents for the current query, we first train personalized word vectors on the user's search history before this query by word2vec model [25] to obtain the query and document representations, and then compute the relevance scores with the KNRM model and SLTB features.

(7) **PWEBA**: This [28] is a personalization model for Twitter search. It first trains personal word embeddings on the user's history and create a word-synonym table based on word vector similarity, then it re-ranks the generic list with cosine similarities between the query's synonyms and documents.

### 4.3 Model Settings

In our model, we initialize the personal word embeddings with a 100-dimensional word2vec model trained with the whole historical and training data for both datasets. For AOL set, the words with less than 5 occurrences and entropy less than 0.7 are filtered from the personal vocabulary. The min occurrences is set as 8 and the word entropy is 0.65 for the commercial dataset. We set the max length of query as 20 for both datasets, the max length of document titles are 50 for the AOL dataset, and the max document length is set as 500 for the commercial dataset. As for the multi-head self-attention, we use 8 heads and the dimension of each head is 50. The KNRM component has 11 kernels with $\mu \in \{-0.9, -0.7, \ldots, 0.9, 1\}$ and $\sigma$ is set to 0.1. The size of hidden states in GRU is 100. Model optimization uses the Adam optimizer, with batchsize as 200, learning rate as 1e-3 and $\epsilon = 1e-5$. We train the model for totally 5-10 epochs and store the one performing best on the validate set.

## 5 EXPERIMENTAL RESULTS AND ANALYSIS

### 5.1 Overall Performance

To begin with, we compare the overall performances of all baselines and PEPS. We train all models on the training set, and then evaluate on the testing set without any update. The reason of using P-Imp only on the commericial dataset has been stated in Section 4.1. Results are shown in Table 2. We find:

(1) **Compared to all the baselines, our PEPS model achieves significant improvements in terms of all the evaluation metrics, with paired t-test at p<0.01 level on the AOL query log and paired t-test at p<0.05 level on the commercial dataset.** Especially for the two state-of-the-art personalized search models HRNN and PSGAN, our model outperforms them greatly. On the AOL set, our model improves PSGAN by 30.9% in MAP and 30.5% in MRR metric. In addition, it promotes 1.1% in MAP and 2.3% in P-Imp which evaluates models from a more credible perspective on the commercial set. Both HRNN and PSGAN models tailor the original document list by building user interest profiles. This results prove that the PEPS model proposed in an alternative way is also effective for personalization and achieves the best performance.

**Table 3: Results of ablation experiments. Relative performances compared with complete PEPS are in percentages. PWE/GWE means personal/global word embeddings.**

| PEPS Variant | AOL Dataset | | | | | | Commercial Dataset | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | | MRR | | P@1 | | MAP | | MRR | | P@1 | | P-Imp. | |
| PEPS | .7174 | - | .7309 | - | .6338 | - | .8221 | - | .8321 | - | .7251 | - | .2545 | - |
| w/o Attn. | .6869 | -4.25% | .7008 | -4.12% | .6021 | -5.00% | .8145 | -0.84% | .8254 | -0.83% | .7196 | -1.18% | .2446 | -4.08% |
| w/o Attn, PWE | .6693 | -6.70% | .6823 | -6.65% | .5771 | -8.95% | .8126 | -1.07% | .8242 | -0.97% | .7181 | -1.39% | .2388 | -6.35% |
| w/o Attn, GWE | .6686 | -6.80% | .6822 | -6.66% | .5796 | -8.55% | .8139 | -0.91% | .8249 | -0.89% | .7191 | -1.25% | .2418 | -5.18% |
| Ablation on query reformulation | | | | | | | | | | | | | | |
| w/o Multi-task | .7161 | -0.18% | .7298 | -0.15% | .6326 | -0.19% | .8186 | -0.34% | .8295 | -0.34% | .7256 | -0.36% | .2513 | -1.45% |
| w/o Query Ref | .7102 | -1.00% | .7235 | -1.01% | .6262 | -1.20% | .8202 | -0.15% | .8306 | -0.20% | .7253 | -0.40% | .2392 | -6.20% |

(2) **Comparing with the closer baseline PPWE, our PEPS model still performs much better whether fix the personal word embeddings or not**. PPWE is our proposed pipeline personalized model which trains fixed personal word vectors with the word2vec for ranking, without supervised finetuning. Compared to PPWE, the end-to-end PEPS introduces the idea of finetuning the pretrained personal word embeddings with the click labels and uses the multi-head self-attention mechanism to capture the interactions between contexts to clarify the personalized meaning of a specific word. The obvious promotion on the PPWE model confirms that it is not only the static individual search logs but also training with the click labels in a supervised way that produces personal word embeddings containing accurate user interests. Furthermore, the contextual information is also important.

(3) **Generally, all personalized search models improve the original ranking results greatly, indicating that personalization is helpful for promoting users' search experience.** The increase of P-Click model validates the effectiveness of refinding behaviors. SLTB model realizes personalization by extracting various interests-related features from the search history. HRNN and PSGAN which both build user interest profiles with a hierarchical RNN achieve great results, and the great performance of PSGAN confirms the importance of high-quality data for the training of personalized models. The word embedding based models PWEBA, PPWE and PEPS also show great improvements. However, we find the PClick and PWEBA perform worse than the neural adhoc-ranking models KNRM and Conv-KNRM. We analyze it may because the ability of deep learning based models is stronger than traditional models, and the word embedding in PWEBA trained on a single user's history is unreliable.

In a word, the overall performances strongly verify that **our PEPS can obtain personal word embeddings that contain accurate user interests and clarify personalized query intents of ambiguous queries to improve personalization.**

## 5.2 Ablation Experiments

The PEPS model includes several main components: the word embedding layer, text representations and the query reformulation module. To figure out the role of each part for personalized search, we perform several ablation experiments. We illustrate the experimental results in Table 3 and make some discussions.

**Personal & global word embeddings** In order to confirm the respective effects of the global and personal word embeddings, we alternatively strip off the two parts to conduct experiments. We also turn off the attention mechanism to make the comparison results more clear. The results are presented at the $3^{rd}$, $4^{th}$ row in Table 3. We find the model loses 6.7% in MAP and 8.95% in P@1 without the personal word vectors on the AOL dataset. As for the commercial set, the performance of PEPS drops 1.07% and 6.35% in the MAP and P-Imp. After removing the global word embeddings, the impacts on the model are similar. This indicates that the personal word embeddings containing user interests is critical for personalization, but every user has changing interests and growing knowledges, and they are likely to use new meanings of a word that has never been involved in their history. In such cases, the global embedding can be helpful to provide general great results.

**Contextual representation** On the top of the word representations, we apply a multi-head self-attention layer to obtain the contextual representations taking the word interactions into account. We disable the attention layer to analyze the contribution of the context, and report the results in Table 3. Without the attention layer, the MAP, MRR, P@1 metrics drop 4.25%, 4.12%, 5.0% on the AOL dataset and 0.84%, 0.83%, 1.18% on the commercial dataset respectively. This demonstrates that the specific meaning of a word not only depends on itself but also the context, thus the multi-head attention contributes to clarify the meaning of a word.

**Query reformulation** We use a query reformulation task to help figure out the user's real query intent through joint learning. To analyze the impacts carefully, we remove the whole query reformulation module or only turn off the joint learning (i.e. the decoder) and report their performance in Table 3. On the AOL log, when only the decoder is removed, there is a slight impact on the results, but the performance drops obviously without the whole module. As for the commercial set, the results of the P@1 and P-Imp metrics also follow this change trend. It illustrates that the personalized query sentence representation is helpful but the contribution of the joint learning is limited. We conjecture it is because the training of the ranking model relies more on the click information.

## 5.3 Experiments on Different Query Sets

To better analyze how our model improves the personalized search, we divide all queries into different sets, and compare the improvement of MAP based on the original ranking result of our PEPS and several baselines. The whole results are illustrated in Figure 2.

**Informational & Navigational Queries** Click entropy is an effective measure of whether a query is ambiguous. Studies [16, 33]

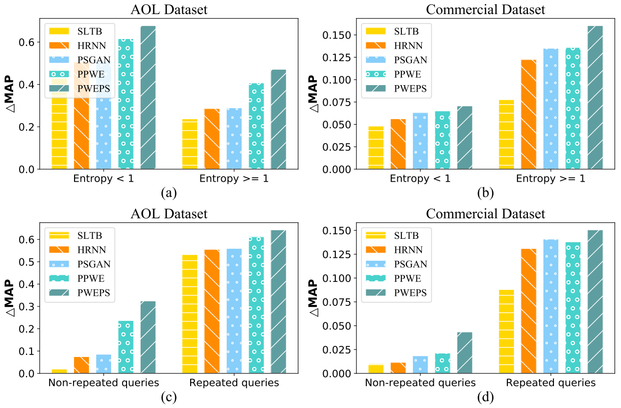

**Figure 2: Experimental results on different query sets. (a) and (b) are results about queries with different entropies, (c) and (d) are results on repeated/non-repeated queries.**

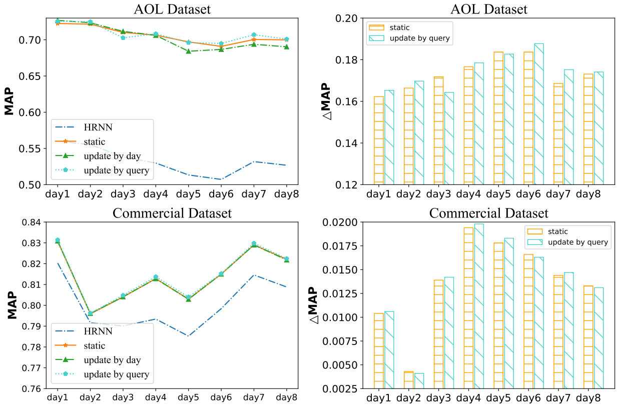

**Figure 3: Performance of different online update methods.**

have shown that it is more necessary to personalize the search results for queries with higher click entropy. Thus, we divide all queries into informational queries with click entropy >= 1 and navigational queries with entropy < 1. Figure 2 (a) and (b) show the performance of our model and baselines on the two datasets.

First, although the relative performance on the two query sets of all models are opposite on the two datasets, which we think may due to the data distribution, we find our model consistently outperforms all baselines on both query groups of the two datasets. Specifically, compared to the best baseline PSGAN, our model also has obvious improvements no matter the query is clear or ambiguous, especially on the informational query set. It confirms the ability of PEPS to perform well on queries that more require personalization.

**Repeated & Non-repeated Queries** In personalized search, the user's behaviors on the relevant queries in the search history are critical to analyze the user's interests for the current query. Some studies [6, 16] even directly use the click features to promote document ranking, but such methods lack the ability of generalization. To further explore the generalization and learning abilities of our model, we categorize all testing queries into repeated or new queries according to whether they have appeared in the search history. The comparison results are shown in Figure 2 (c) and (d).

Consistently, we find all personalized search models achieve greater improvements on the repeated query set for the two datasets. This demonstrates that most personalized models can satisfy the user's refinding needs well, but some may fail on the non-repeated queries, especially the traditional feature-based SLTB model. Our PEPS shows the best results on both query sets and the proportion of improvement on the non-repeated queries is greater, which verifies that our model can not only apply the click-based features to support the user's refindings but also can learn the user's real interests to improve the personalized results of newly issued queries.

## 5.4 Experiments with Online Update Methods

We design three approaches to finetune the word embeddings along with the user's search process to capture the user interests reflected in the newly issued queries. To explore the effects. we perform simulation experiments on the last 8-day testing data. We set a day as a stage and adjust the word embeddings with two approaches. We calculate MAP on the data of each day, obtaining the performance curves and MAP improvements on HRNN shown in Figure 3.

Focusing on the left graphs in Figure 3, we can find that all the performance curves show similar change trends, which should be determined by the distribution of the testing data. And the curves of our model lie above that of HRNN. Comparing the two different adjustment approaches, the method of updating by query outperforms the other on both datasets. It indicates that the short-term user interests in the same session are very effective for improving the results of the subsequent queries. In general, both the update methods improve the MAP compared to the static test in the first several days, but only the update by query method performs better than the static test in the latter days on the AOL set, and the other method is worse. We analyze the possible reason is the incremental finetune approach is unstable and difficult to make the model achieve the global optimal state, and a long-term finetune may make the model perform worse. As for this problem, we also propose a solution that we continuously update the word embeddings in a short time, and then a batch of new samples are added to train a global optimal model after a long time.

# 6 CONCLUSION

In this paper, we implemented search results personalization in an alternative way. Different from existing personalized search approaches which mainly create user profiles and personalize results based on the created profiles, we explore the idea that different users have different understandings of the same word. We proposed PEPS - a model in which we train personal word embeddings for each individual user. Furthermore, we apply self-attention mechanism to obtain the contextual representations for clarifying the query intent. Then, we designed a multi-task framework including personalized ranking and query reformulation to jointly train the personal word embeddings and ranking model. We also worked out three approaches for online update to track the new user interests. Experimental results on two large-scale query logs verified the effectiveness of our model. In the future, we will explore better user interest representation learning algorithms.

# REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-Task Learning for Document Ranking and Query Suggestion. In *6th International Conference on Learning Representations, ICLR 2018,.*

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

[3] Nawal Ould Amer, Philippe Mulhem, and Mathias Géry. 2016. Toward Word Embedding for Personalized Information Retrieval. *CoRR* abs/1606.06991 (2016).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

[5] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and using location metadata to personalize web search. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011.* 135–144.

[6] 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 *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012.* 185–194.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada].* 601–608.

[8] 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 *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005.* 89–96.

[9] Chris J. C. Burges, Krysta M. Svore, Qiang Wu, and Jianfeng Gao. 2008. *Ranking, Boosting, and Model Adaptation.* Technical Report MSR-TR-2008-109. 18 pages.

[10] Fei Cai, Shangsong Liang, and Maarten de Rijke. 2014. Personalized document re-ranking based on Bayesian probabilistic matrix factorization. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014.* 835–838.

[11] Mark James Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards query log based personalization using topic models. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010.* 1849–1852.

[12] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL.* 1724–1734.

[13] Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Sebastian de la Chica, and David Sontag. 2011. Personalizing web search results by reading level. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011.* 403–412.

[14] Nick Craswell, Onno Zoeter, Michael J. Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008.* 87–94.

[15] 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, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018.* 126–134.

[16] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007.*

[17] 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 2018, Torino, Italy, October 22-26, 2018.*

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *CoRR* abs/1406.2661 (2014).

[19] Morgan Harvey, Fabio Crestani, and Mark James Carman. 2013. Building user profiles from topic models for personalised search. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013.* 2309–2314.

[20] Guangneng Hu. 2019. Personalized Neural Embeddings for Collaborative Filtering with Text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers).* 2082–2088.

[21] Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation Inference Network for Context-Aware Query Suggestion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018.* 197–206.

[22] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR.* 154–161.

[23] Cheng Li, Mingyang Zhang, Michael Bendersky, Hongbo Deng, Donald Metzler, and Marc Najork. 2019. Multi-view Embedding-based Synonyms for Email Search. In *Proceedings of SIGIR 2019.* 575–584.

[24] 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 SIGIR 2019.* 555–564.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR 2013, Proceedings.*

[26] Daisuke Oba, Naoki Yoshinaga, Shoetsu Sato, Satoshi Akasaki, and Masashi Toyoda. 2019. Modeling Personal Biases in Language Use by Inducing Personalized Word Embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers).* 2102–2108.

[27] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[28] Sameendra Samarawickrama, Shanika Karunasekera, Aaron Harwood, and Ramamohanarao Kotagiri. 2017. Search Result Personalization in Twitter Using Neural Word Embeddings. In *Big Data Analytics and Knowledge Discovery - 19th International Conference, DaWaK 2017.* 244–258.

[29] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. 2007. Web search personalization with ontological user profiles. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007.*

[30] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep RankNet for personalized search. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014.* 83–92.

[31] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015.* 553–562.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada.* 3104–3112.

[33] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. 2008. To personalize or not to personalize: modeling queries with variation in user intent. In *Proceedings of SIGIR 2008, Singapore, July 20-24, 2008.* 163–170.

[34] Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011.* 85–94.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30.*

[36] Maksims Volkovs. 2015. Context Models For Web Search Personalization. *CoRR* abs/1502.00527 (2015).

[37] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search Personalization with Embeddings. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017.*

[38] Thanh Tien Vu, Alistair Willis, Son Ngoc Tran, and Dawei Song. 2015. Temporal Latent Topic User Profiles for Search Personalisation. In *Advances in Information Retrieval - 37th European Conference on IR Research, ECIR 2015.* 605–616.

[39] Hongning Wang, Xiaodong He, Ming-Wei Chang, Yang Song, Ryen W. White, and Wei Chu. 2013. Personalized ranking model adaptation for web search. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, 2013.*

[40] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. A User-Item Relevance Model for Log-Based Collaborative Filtering. In *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006.* 37–48.

[41] Ryen W. White, Wei Chu, Ahmed Hassan Awadallah, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *22nd International World Wide Web Conference, WWW '13.* 1411–1420.

[42] 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 SIGIR 2017.* 55–64.