Improving Session Search by Modeling Multi-Granularity Historical Query Change

Xiaochen Zuo, Zhicheng Dou, and Ji-Rong Wen
Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
Key Laboratory of Data Engineering and Knowledge Engineering, MOE, China
dou@ruc.edu.cn

ABSTRACT

In session search, it's important to utilize historical interactions between users and the search engines to improve document retrieval. However, not all historical information contributes to document ranking. Users often express their preferences in the process of modifying the previous query, which can help us catch useful information in the historical interactions. Inspired by it, we propose to model historical query change to improve document ranking performance. Especially, we characterize multi-granularity query change between each pair of adjacent queries at both term level and semantic level. For term level query change, we calculate three types of term weights, including the retained term weights, added term weights and removed term weights. Then we perform term-based interaction between the candidate document and historical queries based on the term weights. For semantic level query change, we calculate an overall representation of user intent by integrating the representations of each historical query obtained by different types of term weights. Then we adopt representation-based matching between this representation and the candidate document. To improve the effect of query change modeling, we introduce query change classification as an auxiliary task. Experimental results on AOL and TianGong-ST search logs show that our model outperforms most existing models for session search.

CCS CONCEPTS

• **Information systems** → *Similarity measures*; *Query reformulation.*

KEYWORDS

Session Search, Query Change, Document Ranking

ACM Reference Format:
Xiaochen Zuo, Zhicheng Dou, and Ji-Rong Wen. 2022. Improving Session Search by Modeling Multi-Granularity Historical Query Change. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498415

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498415

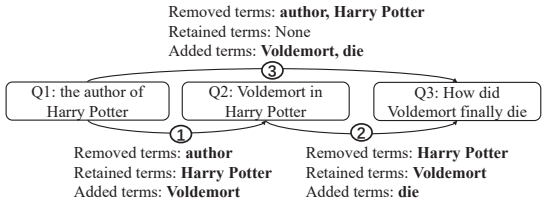


Figure 1: An example to show the potential of modeling historical query change.

1 INTRODUCTION

Users usually put forward a series of queries to solve a search task or multiple similar search tasks [1]. This process can be regarded as an interaction between users and the search engine. During the interactions, search engines can record much information for processing the following user queries. The information includes users' historical queries and satisfying clicks, which helps clarify users' search intent. For example, when a user searches "attractions in Beijing", if he has searched "Beijing traditional snacks" before, it's better to return some attractions with delicious food, while if he has searched "modern history of Beijing", it's better to return some historical sites in Beijing. The example shows historical interactions can provide useful information for document retrieval.

However, not all information in the historical interactions is useful, some information may even mislead the understanding of current user intent. Figure 1 shows an example, where a user has searched "the author of Harry Potter" and "Voldemort in Harry Potter", and now he wants to know "how did Voldemort finally die". It's obvious that Harry Potter's author is not helpful for the current document ranking, which is mentioned in the first query. What's more, if the search engine only focuses on the change between the current query and historical queries (labeled as ② and ③), it probably thinks that "Harry Potter" is not important, since it was removed in the current query. On the contrary, "Harry Potter" is probably omitted by the user for the sake of brevity, and if the search engine observes that the user retains it in the first two queries (labeled as ①), it may increase the importance of "Harry Potter". This example demonstrates the potential of modeling historical query change for the current document ranking. It can highlight useful information more accurately compared to directly making a comparison between the current query and historical queries.

Some traditional methods [19, 32] are proposed to model search session through Markov Decision Process (MDP). They continuously adjust the word weights by modeling the change between historical queries. But these methods adjust the weight based on

the exact match of the word without considering the meaning of the word. Recently, some methods [2, 3, 6, 8, 16, 20, 27, 31, 34] are proposed to model query change, but they focus on modeling the change between the current query and historical queries, and not pay attention to the effect of change between historical queries. Furthermore, most of them adopt representation-based sequential modeling. Specifically, they first formulate the representations of queries and documents and then utilize recurrent neural networks [11] or attention mechanism [28] to model the session process. The advantage of these methods is that they can better model the semantic change of the queries. However, much term level information is lost in the process of representation learning. Besides, a distributed representation of user intent can not reflect which terms in the historical queries are more important for current document ranking. So it's difficult to make fine-grained interaction between historical queries and candidate document, which have been proven [21, 33] to be more effective than most representation-focused models [12, 13].

To solve the shortcomings of existing methods, we propose the Historical Query Change aware ranking Network (HQCN) to improve session search by modeling historical query change. In HQCN, we model historical query change at both term level and semantics level, which are used for multi-granularity interactions between historical queries and document. To be specific, **firstly**, for each pair of adjacent queries in the session, we design a query change modeling unit (QCMU) to measure the term change between the queries. The unit adopts the attention mechanism to calculate three types of term weights: 1) removed term weights for the previous query, 2) retained term weights for the next query, and 3) added term weights for the next query. In order to improve the reliability of these three types of term weights, we designed an auxiliary learning task, namely query change classification, to help training the parameters in the QCMU. **Secondly**, these term weights are utilized to perform term based interaction between historical queries and candidate document through attentive kernel pooling method. **Thirdly**, to model semantic level query change, we adopt Transformers [28] to integrate the representations of each historical query calculated by different types of term weights and formulate an overall representation of user intent. Then we perform representation based matching between the candidate document and user intent. **Fourthly**, we combine the context-aware ranking scores obtained by term-based interaction and representation-based matching and get the final document ranking score. We perform experiments on AOL search logs [22] and TianGong-ST [5] search logs datasets. The results show that our model outperforms most existing neural ranking models for session search. Furthermore, compared to HBA-Transformers [23], our model greatly improves the computational efficiency and achieves better results.

The contributions of this paper can be summarized as follows: 1) We propose to improve session search by modeling historical query change, which is effective to highlight useful information in the search context. 2) We analyse multi-granularity historical query change at both term level and semantic level, which are separately used for term based interaction and representation based matching between historical queries and candidate documents. 3) We propose to employ query change classification as an auxiliary learning task to effectively model term-level query change.

2 RELATED WORK

Most of the existing session search model focused on user intent modeling. Some previous works regard user's search intent as implicit variables and try to estimate them. E.g. Shen et al. [25] utilized historical queries and clicked documents to estimate two language models, and then used the Kullback-Leibler divergence to measure the relevance of candidate documents. Cao et al [4]. employed HMM to model the process of session search. With the widespread application of deep neural networks, more and more models are proposed to model session context to obtain semantic representation of users' search intent. Sordoni et al. [27] used the hierarchical recurrent encoder to obtain the hidden representation of the user's intent for each historical query. Similarly, Wu et al. [31] introduced user feedback information to the hidden representation in the recurrent neural networks. Wasi et al. [2, 3] adopt a similar model structure, and they utilized a multi-task learning strategy to train the model for query suggestion and document ranking. Halder et al. [9] modeled the information deficit by analyzing each historical queries-clicks pair. Qu et al.[23] designed a hierarchical transformers-based model to distinguish different user behaviors.

Besides, some existing methods also try to model query change to improve session search, which can be divided into two categories: By modeling 1) term-level query change and 2) semantic-level query change. To model term-level query change, Guan et al. [7] proposed a query change retrieval model (QCM). In QCM, the search engine agent increases or decreases a term's weight according to its type and its appearance in the previous user clicks. Xiang et al. [32] proposed four principles to characterize query change and designed ranking features according to these principles. Sloan et al. [26] proposed to make inferences on the user's information need by understanding how a user's queries evolve throughout the session. The main drawback of these works is that they increased or decreased the weights of terms separately, which ignored the overall semantics of the query. For semantic-level query change modeling, Jiang et al. [16] obtained the query change representation through the difference between the embeddings of the current query and the next query. He et al.[10] proposed to bridge the gap between web documents and user queries by query rewriters. Mitra [20] calculated the distributed representations of query change. Verma et al. [29] utilized the generation-based sequence-to-sequence models that capture session context to obtain the reformulated query. These query reformulation models focused on learning a representation of user intent or rewrite a new query, without considering to utilize session context directly.

3 OUR APPROACH

In this section, we describe the architecture of our proposed model HQCN. We first make a definition of the session search task, then we introduce four components in the model: query term weighting, representation-based matching, term-based interaction and document scoring.

3.1 Problem Definition

In session search, the search engine retrieves a ranking list of candidate documents according to a given query and the session context. The session context contains the historical queries and the clicks for

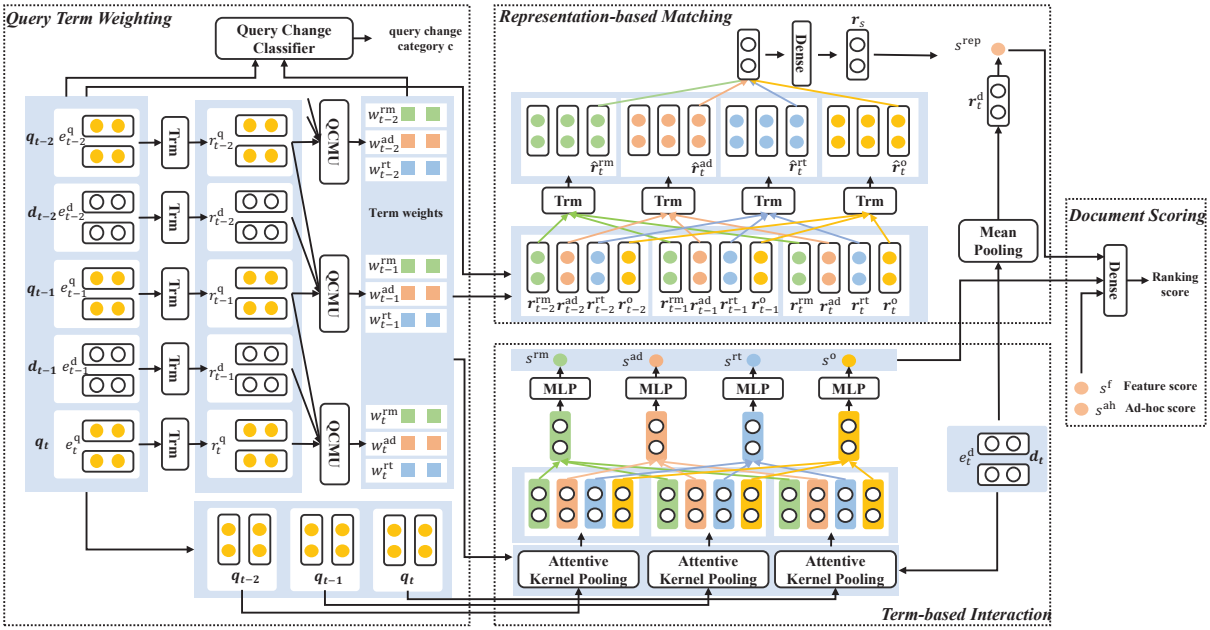


Figure 2: The overall structure of Historical Query Change Aware Ranking Network (HQCN).

each query. Suppose the given query is q_t, and the session context is $\mathcal{S}$, which is defined as: $\mathcal{S} = [(q_1, d_1), ..., (q_{t-1}, d_{t-1})]$. q_i is the i-th query in the session and d_i is the satisfied click (the highest ranked click document) for query q_i. The task is to rank the candidate documents in $\mathcal{D}_t$ for query q_t. Specifically, for each candidate d_t in $\mathcal{D}_t$, we calculate its ranking score $\mathcal{R}(d_t|q_t, \mathcal{S})$ and obtain the ranking list according to the score.

3.2 HQCN: Historical Query Change Aware Ranking Network

The overall structure of HQCN is shown in Figure 2. The model can be divided into four parts: query term weighting, representation-based matching, term-based interaction, and document scoring. We first utilize query term weighting to model historical query change and calculate query term weights. Query change classification is also implemented in this part. Then we use the term weights to calculate the representation of user intent and perform representation-based matching with the candidate document. After that, we also used the term weights to perform term-based interaction between candidate document and each historical query. Then we calculate a final ranking score for the candidate document.

(1) **Query term weighting**. In this part, we calculate the weights of terms in the queries based on term-level query change. We describe the term-level query change in three aspects: removed terms from the previous query, retained and added terms in the next query. First of all, we obtain the contextual word representations of all queries and clicks through Transformers [28]:

$$\begin{aligned}\boldsymbol{r}^{\mathrm{q}}_{i,1}, ..., \boldsymbol{r}^{\mathrm{q}}_{i,|q_i|} &= \mathrm{Trm}\left(\boldsymbol{e}^{\mathrm{q}}_{i,1} + \boldsymbol{e}^{\mathrm{p}}_{1}, ..., \boldsymbol{e}^{\mathrm{q}}_{i,|q_i|} + \boldsymbol{e}^{\mathrm{p}}_{|q_i|}\right), \quad i \in [1, t].\\ \boldsymbol{r}^{\mathrm{d}}_{j,1}, ..., \boldsymbol{r}^{\mathrm{d}}_{j,|d_j|} &= \mathrm{Trm}\left(\boldsymbol{e}^{\mathrm{d}}_{j,1} + \boldsymbol{e}^{\mathrm{p}}_{1}, ..., \boldsymbol{e}^{\mathrm{d}}_{j,|d_j|} + \boldsymbol{e}^{\mathrm{p}}_{|d_j|}\right), \quad j \in [1, t-1].\end{aligned} \tag{1}$$

Where $\boldsymbol{e}^{\mathrm{q}}_i$ and $\boldsymbol{e}^{\mathrm{d}}_j$ are embeddings of words in query q_i and the satisfied click d_j, $\boldsymbol{e}^{\mathrm{p}}$ is the position embeddings. To describe the query change between q_i and q_{i-1}, we calculate three types of term weights: the removed term weights $\boldsymbol{w}^{\mathrm{rm}}_{i-1}$ for query q_{i-1}, the added term weights $\boldsymbol{w}^{\mathrm{ad}}_i$ and the retained term weights $\boldsymbol{w}^{\mathrm{rt}}_i$ for query q_i. When calculating the added and retained term weights $\boldsymbol{w}^{\mathrm{ad}}_i$ and $\boldsymbol{w}^{\mathrm{rt}}_i$, we also consider the satisfied click d_{i-1}, since users are likely to add or retain some words according to it. We obtain the three types of term weights through a query change modeling unit (QCMU):

$$\boldsymbol{w}^{\mathrm{rm}}_{i-1}, \boldsymbol{w}^{\mathrm{ad}}_i, \boldsymbol{w}^{\mathrm{rt}}_i = \mathrm{QCMU}\left(\boldsymbol{r}^{\mathrm{q}}_{i,\cdot}, \boldsymbol{r}^{\mathrm{q}}_{i-1,\cdot}, \boldsymbol{r}^{\mathrm{d}}_{i-1,\cdot}\right). \tag{2}$$

Specifically, to obtain the retained weights $\boldsymbol{w}^{\mathrm{rt}}_i$ for terms in query q_i, we compare the contextual word representations in q_i with those in q_{i-1} and d_{i-1}. First, we concatenate the word representations in q_{i-1} and d_{i-1} to obtain the history sequence H:

$$\begin{aligned}H &= \left[\boldsymbol{h}_1, ..., \boldsymbol{h}_{|q_{i-1}|+|d_{i-1}|}\right]\\ &= \left[\boldsymbol{r}^{\mathrm{q}}_{i-1,1}, ..., \boldsymbol{r}^{\mathrm{q}}_{i-1,|q_{i-1}|}\boldsymbol{r}^{\mathrm{d}}_{i-1,1}, ..., \boldsymbol{r}^{\mathrm{d}}_{i-1,|d_{i-1}|}\right].\end{aligned}$$

Then we perform interaction between word representations in q_i and H to get the relevance score v_k for the k-th word in query q_i:

$$v^{\mathrm{rt}}_k = \sum_{l=1}^{|H|} m^{\mathrm{rt}}_{k,l}, \quad m^{\mathrm{rt}}_{k,l} = \frac{\left(\boldsymbol{W}^{\mathrm{rt}}_1 \boldsymbol{r}^{\mathrm{q}}_{i,k}\right)^T \boldsymbol{W}^{\mathrm{rt}}_2 \boldsymbol{h}_l}{\sqrt{d}}. \tag{3}$$

We normalize the relevance score through softmax function to obtain the retained term weights $\boldsymbol{w}^{\mathrm{rt}}_i = \mathrm{softmax}\left(\boldsymbol{v}^{\mathrm{rt}}\right)$.

Similarly, to calculate the added term weights for query q_i, we adopt the same method in Equation (3) to get the relevance score v^{ad}_k for the k-th term, but we use different parameters $\boldsymbol{W}^{\mathrm{ad}}_1$ and $\boldsymbol{W}^{\mathrm{ad}}_2$. Differently, the added terms in q_i are less related to the terms in

q_{i-1} and d_{i-1}. So when doing softmax, we take a negative value for each relevance score, i.e., we let $\boldsymbol{w}_i^{\text{ad}} = \text{softmax}\left(-\boldsymbol{v}^{\text{ad}}\right)$.

As for the removed weights for terms in q_{i-1}, we only have to compare the difference between q_{i-1} and q_i. We directly make interaction between the contextual word representations of q_i and q_{i-1} to obtain the relevance score v_k^{rm} for the k-th word in q_{i-1}:

$$v_k^{\text{rm}} = \sum_{l=1}^{|q_i|} m_{k,l}^{\text{rm}}, \quad m_{k,l}^{\text{rm}} = \frac{\left(\boldsymbol{W}_1^{\text{rm}} \boldsymbol{r}_{i-1,k}^{\text{q}}\right)^T \boldsymbol{W}_2^{\text{rm}} \boldsymbol{r}_{i,l}^{\text{q}}}{\sqrt{d}}. \tag{4}$$

Then we use the same normalization strategy as added term weights to take a negative value for each relevance score v_k^{rm} before the softmax function, i.e., we have $\boldsymbol{w}_i^{\text{rm}} = \text{Softmax}\left(-\boldsymbol{v}^{\text{rm}}\right)$.

In order to make these three types of word weights learn their corresponding meanings better, we adopt query change classification as an auxiliary learning task. Inspired by existing works [14, 15], we propose to divide query change into four categories: **generalization**, **exploitation**, **exploration**, and **new task**. Generalization means the next query is more general than the previous query. Exploitation means the next query is more specific than the previous query. Exploration means the two adjacent queries are generally related but describe different subtopics. New task means the two adjacent queries describe completely different topics.

Given two adjacent queries, we humans usually judge the change category by comparing the similarities and differences between the two queries. Specifically, if the next query doesn't add any information but removes some information in the previous query, it's likely to be generalization. If the next query doesn't remove any information but adds some information, it's likely to be exploitation. If the next query retains some information and replaces some information, it tends to be exploration. If the next query removes all information and adds some new information, it's likely to be a new task. In summary, the query change categories can be described by the retained, added, and removed information of queries.

Based on the analyse above and the term weights calculated in Eq. 2, we propose a Query Change Classifier (QCC) to classify each pair of adjacent queries into four query change categories:

$$\hat{c} = \arg\max_{c \in C} p(c|q_i, q_{i-1}, d_{i-1}) = \text{QCC}\left(q_i, q_{i-1}, d_{i-1}\right) \tag{5}$$

In QCC, we first calculate three types of query representations: $\boldsymbol{r}_{i-1}^{\text{rm}}$, $\boldsymbol{r}_i^{\text{ad}}$ and $\boldsymbol{r}_i^{\text{rt}}$ for an adjacent query pair $<q_{i-1}, q_i>$:

$$\boldsymbol{r}_{i-1}^{\text{rm}} = \sum_k \boldsymbol{w}_{i-1,k}^{\text{rm}} \boldsymbol{e}_{i-1,k}^{\text{q}}, \quad \boldsymbol{r}_i^{\text{ad}} = \sum_k \boldsymbol{w}_{i,k}^{\text{ad}} \boldsymbol{e}_{i,k}^{\text{q}}, \quad \boldsymbol{r}_i^{\text{rt}} = \sum_k \boldsymbol{w}_{i,k}^{\text{rt}} \boldsymbol{e}_{i,k}^{\text{q}}. \tag{6}$$

Then we make comparison between $\boldsymbol{r}_{i-1}^{\text{rm}}$ with the representation of q_{i-1} to extract the removed information of q_{i-1}:

$$\Delta \boldsymbol{r}_{i-1}^{\text{rm}} = \boldsymbol{r}_{i-1}^{\text{rm}} - \frac{1}{|q_{i-1}|} \sum_{j=1}^{|q_{i-1}|} \boldsymbol{e}_{i-1,j}^{\text{q}}.$$

Similarly, to extract the added and retained information of q_i, we compare $\boldsymbol{r}_i^{\text{ad}}$ and $\boldsymbol{r}_i^{\text{rt}}$ with the representation of q_i separately:

$$\Delta \boldsymbol{r}_i^{\text{ad}} = \boldsymbol{r}_i^{\text{ad}} - \frac{1}{|q_i|} \sum_{j=1}^{|q_i|} \boldsymbol{e}_{i,j}^{\text{q}}, \quad \Delta \boldsymbol{r}_i^{\text{rt}} = \boldsymbol{r}_i^{\text{rt}} - \frac{1}{|q_i|} \sum_{j=1}^{|q_i|} \boldsymbol{e}_{i,j}^{\text{q}}.$$

Finally, we concatenate these compared representations to make query change classification:

$$\begin{aligned} p(c|q_i, q_{i-1}, d_{i-1}) &= \frac{\exp \boldsymbol{s}_c}{\sum_{c' \in C} \exp \boldsymbol{s}_{c'}}, \\ \boldsymbol{s} &= \text{Dense}\left(\left[\Delta \boldsymbol{r}_{i-1}^{\text{rm}}; \Delta \boldsymbol{r}_i^{\text{ad}}, \Delta \boldsymbol{r}_i^{\text{rt}}\right]\right), \end{aligned} \tag{7}$$

where $\boldsymbol{s}$ is a 4-d vector, and $\boldsymbol{s}_c$ indicates the score of category c.

(2) **Representation-based matching**. In this part, we model semantic-level query change and learn the representation of user intent, which is utilized to perform matching with the candidate document. In the query term weighting part, we have obtained three types of representations for each query: $\boldsymbol{r}_i^{\text{rm}}$, $\boldsymbol{r}_i^{\text{ad}}$ and $\boldsymbol{r}_i^{\text{rt}}$ according to Eq. 6. Then we add an original query representation $\boldsymbol{r}_i^{\text{o}} = \sum_j \boldsymbol{e}_{i,j}^{\text{q}}$ that contains the general information of each query.

To model the process of query change on the semantic level, we have to make interactions between the queries in the session. Based on different aspects of query representations, we can model query change from different perspectives. We adopt Transformers [28] to fulfill the interactions between queries. Specifically, for the removed query representations $\boldsymbol{r}^{\text{rm}}$ of all the queries, we adopt Transformers to obtain the interacted representations:

$$\hat{\boldsymbol{r}}_1^{\text{rm}}, ..., \hat{\boldsymbol{r}}_t^{\text{rm}} = \text{Trm}\left(\boldsymbol{r}_1^{\text{rm}} + \boldsymbol{e}_1^{\text{p}}, ..., \boldsymbol{r}_t^{\text{rm}} + \boldsymbol{e}_t^{\text{p}}\right). \tag{8}$$

Here $\boldsymbol{e}^{\text{p}}$ is the same position embeddings as that in Eq. (1). We use the same method to obtain other types of query representations $\hat{\boldsymbol{r}}^{\text{ad}}$, $\hat{\boldsymbol{r}}^{\text{rt}}$ and $\hat{\boldsymbol{r}}^{\text{o}}$. We adopt the last query's representation as user's intent, and the overall representation $\boldsymbol{r}_s$ is obtained from four types of query representations by using a dense layer:

$$\boldsymbol{r}_s = \text{dense}\left(\left[\hat{\boldsymbol{r}}_t^{\text{rm}}, \hat{\boldsymbol{r}}_t^{\text{ad}}, \hat{\boldsymbol{r}}_t^{\text{rt}}, \hat{\boldsymbol{r}}_t^{\text{o}}\right]\right). \tag{9}$$

Then we get the matching score s^{rep} by calculating the similarity between $\boldsymbol{r}_s$ and the candidate representation $\boldsymbol{r}_t^{\text{d}} = \sum_j \boldsymbol{e}_{t,j}^{\text{d}}$.

(3) **Term-based interaction**. In this part, we make fine-grained interaction between each query and candidate document based on the term weights calculated in Eq. 2. We introduce the attention mechanism to the existing kernel pooling method in KNRM [33] and propose the attentive kernel pooling. Given term embeddings of a query and a document, we first calculate a matching matrix $\boldsymbol{M}$ according to the similarities between them: $M_{i,j} = \text{cosine}\left(\boldsymbol{e}_i^{\text{q}}, \boldsymbol{e}_j^{\text{d}}\right)$. Then we use attentive kernel pooling to get a k-dimensional soft-tf feature $\boldsymbol{f} = \{K_1(\boldsymbol{M}), ..., K_k(\boldsymbol{M})\}$:

$$K_k\left(\boldsymbol{M}\right) = \sum_i \beta_{i,k} \log \sum_j \exp(-\frac{\left(\boldsymbol{M}_{i,j} - \mu_k\right)^2}{2\sigma_k^2}), \tag{10}$$

where μ_k and σ_k are parameters for the k-th kernel. Each kernel K_k calculates the soft term frequency based on different Gaussian distributions. $\beta_{i,k}$ is the attention coefficient for the i-th word in the query for the k-th kernel, which is calculated by performing the same kernel method on the query term weights $\boldsymbol{w}$:

$$\beta_{i,k} = \exp(-\frac{\left(\boldsymbol{w}_i - \mu_k\right)^2}{2\sigma_k^2}).$$

The rationality of this processing method is that the matrices calculated by the similarity matrix $\boldsymbol{M}$ through different kernels are based on different Gaussian distributions. So we have to calculate

different attention coefficients for features obtained from different kernels. We regard this process as: $\boldsymbol{f} = \text{AKP}(q, d, \boldsymbol{w})$. Where AKP is short for **A**ttentive **K**ernel **P**ooling. Through AKP, we obtain three types of ranking features for each historical query q_i as follows:

$$\begin{aligned} \boldsymbol{f}_i^{\text{rm}} = \text{AKP}\left(q_i, d_t, \boldsymbol{w}_i^{\text{rm}}\right), \ \boldsymbol{f}_i^{\text{ad}} = \text{AKP}\left(q_i, d_t, \boldsymbol{w}_i^{\text{ad}}\right), \\ \boldsymbol{f}_i^{\text{rt}} = \text{AKP}\left(q_i, d_t, \boldsymbol{w}_i^{\text{rt}}\right). \end{aligned} \tag{11}$$

Besides, we also adopt a vanilla kernel pooling method in KNRM to calculate an original query ranking features:

$$\boldsymbol{f}_i^{\text{o}} = \text{Kernel\_Pooling}\left(q_i, d_t\right). \tag{12}$$

The only difference in the vanilla kernel pooling method is that it doesn't have the weight β in Eq. 10. Each type of ranking features $\boldsymbol{f}_i$ is used to calculate the ranking scores through a dense layer:

$$\begin{aligned} s_i^{\text{rm}} = \text{dense}\left(\boldsymbol{f}_i^{\text{rm}}\right), \ s_i^{\text{ad}} = \text{dense}\left(\boldsymbol{f}_i^{\text{ad}}\right), \\ s_i^{\text{rt}} = \text{dense}\left(\boldsymbol{f}_i^{\text{rt}}\right), \quad s_i^{\text{o}} = \text{dense}\left(\boldsymbol{f}_i^{\text{o}}\right). \end{aligned} \tag{13}$$

After obtaining the ranking score for each query, we have to calculate each query's weight to combine the ranking scores. Each query has a different effect on candidate document ranking for the current query. Some queries are important due to the removed terms since they show what the user dislikes. Some queries are important due to the added terms, which reveal the specific preference of the user. And some queries are important due to the retained terms, these terms reflect the general topics that the user is concerned about. So we calculate three different types of query weights: removed query weights $\boldsymbol{a}^{\text{rm}}$, added query weights $\boldsymbol{a}^{\text{ad}}$ and retained query weights $\boldsymbol{a}^{\text{rt}}$. Additionally, we calculate general query weights $\boldsymbol{a}^{\text{o}}$ to combine features $\boldsymbol{f}_i^o$. The weights of the queries are calculated according to their relatedness to the current query. Formally, we adopt the attention mechanism to measure the relatedness:

$$a_k^{\text{rm}} = \frac{\boldsymbol{u}_k^{\text{rm}}}{\sum_i^t \boldsymbol{u}_i^{\text{rm}}}, \ \ \boldsymbol{u}_k^{\text{rm}} = \frac{\left(\boldsymbol{W}_1^{\text{q}} \hat{\boldsymbol{r}}_k^{\text{rm}}\right)^T \boldsymbol{W}_2^{\text{q}} \hat{\boldsymbol{r}}_t^{\text{rm}}}{\sqrt{d}}, \tag{14}$$

where $\boldsymbol{W}_1^{\text{q}}$ and $\boldsymbol{W}_2^{\text{q}}$ are parameters to perform linear transformation on the representations. d is the dimension of the representations. Similarly, we obtain the added query weights $\boldsymbol{a}^{\text{ad}}$, the retained query weights $\boldsymbol{a}^{\text{rt}}$ and the general query weights $\boldsymbol{a}^{\text{o}}$ in the same way. These weights are used to calculate term-level ranking scores:

$$\begin{aligned} s^{\text{rm}} = \sum_{i=1}^{t} a_i^{\text{rm}} s_i^{\text{rm}}, \quad s^{\text{ad}} = \sum_{i=1}^{t} a_i^{\text{ad}} s_i^{\text{ad}}, \\ s^{\text{rt}} = \sum_{i=1}^{t} a_i^{\text{rt}} s_i^{\text{rt}}, \qquad s^{\text{o}} = \sum_{i=1}^{t} a_i^{\text{o}} s_i^{\text{o}}. \end{aligned} \tag{15}$$

(4) **Document Scoring** Finally, we calculate the ranking score $\mathcal{R}\left(d_t | q_t, \mathcal{S}\right)$ by performing a dense layer on the seven scores:

$$\mathcal{R}\left(d_t | q_t, \mathcal{S}\right) = \text{Tanh}\left(\text{dense}\left[s^{\text{rep}}, s^{\text{rm}}, s^{\text{ad}}, s^{\text{rt}}, s^{\text{o}}, s^{\text{ah}}, s^{\text{f}}\right]\right), \tag{16}$$

where s^{ah} is the ad-hoc ranking score obtained through KNRM model. s^{f} is calculated based on adhoc relevance features, including the tf-idf of terms in the candidate, the number of common terms in the query and candidate, and the similarity between the embedding of each query and the candidate.

3.3 Model Training

In the training process, we adopt a multi-task learning strategy to train document ranking and query change models at the same time. For each session $\mathcal{S}$ in the training set, we treat queries other than the first one as the current query, then rank the candidate documents and classify the change type for this query.

Particularly, for the document ranking task, we use the margin ranking loss (margin=1) to train our model:

$$\mathcal{L}_r = \sum_{q_i \in \mathcal{S} \setminus \{q_1\}} \sum_{(d^+, d^-) \in \mathcal{D}_{q_i}^{+,-}} \max\left(0, 1 + \mathcal{R}\left(d^- | q_i\right) - \mathcal{R}\left(d^+ | q_i\right)\right),$$

where $\mathcal{R}\left(d|q\right)$ is short for $\mathcal{R}\left(d|q, \mathcal{S}\right)$. $\mathcal{D}_{q_i}^{+,-}$ contains candidate documents pair for query q_i. For pair $\left(d^+, d^-\right)$, d^+ is clicked by user whereas d^- is not.

For the query change classification auxiliary task, we adopt cross entropy loss for this multi-classification problem:

$$\mathcal{L}_c = \sum_{q_i \in \mathcal{S} \setminus \{q_1\}} -\sum_{c \in C} y_c \log\left(p\left(c | q_i, q_{i-1}, d_{i-1}\right)\right),$$

where y is a one-hot vector indicating the labeled change category. The method to get the label will be introduced in Section 4.2.3. The overall loss is the combination of the two losses:

$$\mathcal{L} = \gamma \mathcal{L}_r + (1 - \gamma)\, \mathcal{L}_c.$$

We train the parameters in HQCN by Back-propagation algorithm and adopt Adam [18] as the optimizer.

4 EXPERIMENTS

4.1 Dataset

We evaluate our proposed method on two public datasets: AOL search logs [22] and Tiangong-ST query logs [5]. MS MARCO Conversational Search Session dataset [1] can also be used for this task. But the sessions in it are constructed manually, not from the real search logs. To analyze the real query change behavior of users, we don't use this dataset. When using AOL and Tiangong-ST datasets, we rank the candidate documents of each query except the first one in the session. The reason is that our method aims to model historical query change. But for the first query in the session, no historical query is available, which will cause most of the parameters in the model unable to be trained. The statistics of the two datasets are shown in Table 1.

When using the AOL search logs, we used the same dataset as constructed in [3]. All the candidates are retrieved by BM25 [24]. The other dataset Tiangong-ST[5] is collected from a Chinese commercial search engine, and it contains web search session data extracted from an 18-day search log. In the training set and validation set, we utilize the clicked documents as the satisfied clicks. In the test set, the candidate documents of each session's last query are manually annotated with a relevance score from 0 to 4. As a result, when evaluating the models on Tiangong-ST, we divide the test set into two parts: 1) **Tiangong-ST Manual**, containing each session's last query with manual labeled relevance scores for candidate documents. 2) **Tiangong-ST Click**, containing other queries with click label as pseudo annotation.

[1] https://github.com/microsoft/MSMARCO-Conversational-Search

Table 1: Statistics of datasets. SL is the length of sessions, QL is the length of queries, DL is the length of document titles, DN is the number of candidate documents, and CN is the number of clicks for each query.

Dataset	AOL			Tiangong-ST		
	train	val.	test	train	val.	test
#Sessions	219,748	34,090	29,369	143,155	2,000	2,000
#Queries	566,967	88,021	76,159	344,806	5,026	6,420
Av.SL	2.581	2.582	2.593	2.409	2.513	3.210
Av.QL	2.862	2.851	2.900	2.894	1.830	3.460
Av. DL	7.27	7.29	7.08	8.25	6.99	9.18
Av.DN	5	5	50	10	10	10
Av.CN	1.084	1.081	1.111	0.937	0.525	3.650

4.2 Experimental Settings

4.2.1 Evaluation metrics. When evaluating the model on the AOL dataset and Tiangong-ST Click, we use MAP, MRR and NDCG [30] as evaluation metrics. To be specific, NDCG includes NDCG@1, NDCG@3, NDCG@5, and NDCG@10. When using the Tiangong-ST Manual test set, we only use NDCG@1, NDCG@3, NDCG@5 and NDCG@10 to evaluate the model, because the relevance label on the test set has five levels of 0-4, which is not suitable to use MAP and MRR because it's unable to determine a score to be the boundary between relevant and irrelevant documents.

4.2.2 Baselines. We use 8 baseline models as comparisons to verify the effect of our model, including 1) **ARC-I** [12], it obtains the representation of each piece of text by two layers of 1-D convolution network. 2) **ARC-II** [12], its utilize 2D-convolution network on the interaction matrix of the query and the document. 3) **KNRM** [33], it performs fine-grained interaction between the current query and candidate document to obtain a matching matrix and adopts kernel pooling method to obtain the ranking features. 4) **Duet** [21], it integrates the representation-focused method and interaction-focused method. 5) **M-NSRF** and 6)**M-Match** [2], they propose to solve query suggestion and document ranking by multi-task learning. 7) **CARS** [3], it also adopts the multi-task learning strategy to solve query suggestion and document ranking at the same time, additionally, it uses attention mechanisms and introduces user clicks to obtain a better representation of session context. 8) **HBA-Transformers** [23], it utilizes hierarchical attention mechanisms to model different user behaviors based on the output contextual representations from Bert.

4.2.3 Annotation for query change classification. As the definition in Section 3.2, there are four types of change between a pair of adjacent queries: generalization, exploitation, exploration, and new task. We design some rules to annotate the change category of each adjacent query pair: 1) If there is no common word in the two queries, we annotate it as "new task". 2) If the second query contains all words in the first query, we annotate it as "exploitation". 3) If the first query contains all words in the second query, we annotate it as "generalization". 4) If none of the above three conditions are satisfied, we annotate it as "exploration". This annotating method is not completely accurate. However, our goal is not to obtain accurate classification results, but to improve the effectiveness of learning term weights in the model by the weakly supervised learning strategy. The effectiveness of the introduction of this auxiliary task will be verified in Section 4.4

4.2.4 Model settings. The dimension of word embedding, position embeddings, and all the intermediate representations is set to 256. We pre-train the word embedding through Fasttext [17], and the training corpus is constructed by all queries and documents in the training set of AOL and Tiangong-ST search logs. The position embeddings in the model are initialized the same as that in [28]. In the kernel pooling layer, we use 11 kernels with different parameters, and one of them is used for exact matching ($\mu = 1.0$ and $\sigma = 0.001$). The hyperparameter γ for multi-task learning is selected from 0.5, 0.8, 0.9, and 0.95. Finally, we find 0.8 has the best performance. We set the dropout rate to 0.1 to avoid over-fitting, and adopt the warm-up strategy with a warm-up portion 10% to dynamically adjust the learning rate. The maximum learning rate is set to 0.0001. We train our model with a 12G TITAN V GPU with the training batch size to 64. It takes about 50 minutes to train an epoch on the AOL search logs dataset and about 40 minutes to train an epoch on the Tiangong-ST dataset[2].

4.3 Overall results

We compare our model HQCN with other baseline models on the two datasets. The results are shown in Table 2 [3]. We use KNRM and HBA-Transformers as representatives to perform the t-test. We have the following findings:

1) **HQCN outperforms all the session search models, which indicates the effectiveness of modeling historical query change**. Compared to the session search models based on multi-task learning M-NSRF, M-Match and CARS, HQCN significantly improves the ranking performance on the two datasets. This proves the effectiveness of historical query change modeling and fine-grained interaction between each query and documents. Compared to the state-of-art session search model HBA-Transformers that utilizes parameters from pre-trained language model, our model still has significant improvement on AOL search logs and Tiangong-ST Click dataset. The reason is that HQRN make interaction between each query and candidate document directly based on the term weights obtained by historical query change modeling. On the contrary, HBA-Transformers generally models the entire sequence without focusing on the interaction of candidate documents with other information. In addition, it is worth to be emphasized that HQCN achieves better results than HBA-Transformers without using additional corpus, and also greatly improves computational efficiency.

2) **HQCN outperforms all the ad-hoc ranking models, which indicates the importance session context modeling**. Compared to the representation focused model ARC-I, interaction focused models ARC-II and KNRM, and Duet that combines these two methods, HQCN achieves significant improvement. These results show the importance of modeling session context including historical queries and clicked documents. However, we find some context-aware ranking models perform worse than some ad-hoc search

[2]Source codes are available at https://github.com/blakezuo/HQCN/tree/master
[3]The results of CARS are lower than that reported in [3]. Because we use different evaluation scripts. We use trec_eval while [3] uses the self-implemented scripts.

Table 2: Overall results on AOL and Tiangong-ST datasets. ⋆ indicates the result is significantly better than KNRM. ⋄ indicates the result is significantly better than HBA-Transformers ($p < 0.05$ in two-tailed paired t-test). Improv. indicates the improvement of HQCN compared to the HBA-Transformers.

Dataset	Metric	ARC-I	ARC-II	KNRM	Duet	M-NSRF	M-Match	CARS	HBA	HQCN	Improv.
AOL Search Logs	MAP	0.3361	0.3834	0.3841	0.4008	0.4217*	0.4459*	0.4297*	0.5281*	0.5448$^{\star}_{\diamond}$	3.16%
	MRR	0.3475	0.3951	0.3933	0.4111	0.4326*	0.4572*	0.4408*	0.5384*	0.5549$^{\star}_{\diamond}$	3.06%
	NDCG@1	0.1988	0.2428	0.2203	0.2492	0.2737*	0.3020*	0.2816*	0.3773*	0.3990$^{\star}_{\diamond}$	5.75%
	NDCG@3	0.3108	0.3561	0.3636	0.3822	0.4025*	0.4301*	0.4117*	0.5241*	0.5441$^{\star}_{\diamond}$	3.82%
	NDCG@5	0.3489	0.4026	0.4109	0.4246	0.4458*	0.4697*	0.4542*	0.5624*	0.5783*	2.83%
	NDCG@10	0.3953	0.4486	0.4574	0.4675	0.4886*	0.5103*	0.4971*	0.5951*	0.6070*	2.00%
Tiangong-ST Click	MAP	0.6597	0.6729	0.6733	0.6622	0.6836	0.6778	0.6909*	0.6957*	0.7389$^{\star}_{\diamond}$	6.21%
	MRR	0.6826	0.6954	0.6932	0.6904	0.7065	0.6993	0.7134*	0.7171*	0.7498$^{\star}_{\diamond}$	4.56%
	NDCG@1	0.5315	0.5458	0.5356	0.5563	0.5609*	0.5499	0.5677*	0.5726*	0.6218$^{\star}_{\diamond}$	8.59%
	NDCG@3	0.6383	0.6553	0.6642	0.6402	0.6698	0.6636	0.6764	0.6807*	0.7315$^{\star}_{\diamond}$	7.26%
	NDCG@5	0.6946	0.7086	0.7104	0.6846	0.7188	0.7199	0.7271	0.7292	0.7643$^{\star}_{\diamond}$	4.81%
	NDCG@10	0.7509	0.7608	0.7609	0.7528	0.7691	0.7646	0.7746	0.7781	0.7924*	1.84%
Tiangong-ST Manual	NDCG@1	0.7088	0.7131	0.7560	0.7120	0.7124	0.7311	0.7385	0.7612	0.7739*	1.67%
	NDCG@3	0.7087	0.7237	0.7457	0.7116	0.7308	0.7233	0.7386	0.7518	0.7682*	2.18%
	NDCG@5	0.7317	0.7379	0.7716	0.7390	0.7489	0.7427	0.7512	0.7639	0.7783	1.89%
	NDCG@10	0.8691	0.8732	0.8894	0.8705	0.8795	0.8801	0.8837	0.8896	0.8976	0.90%

models, e.g. M-NSRF, M-Match and CARS are worse than KNRM on manual annotated Tiangong-ST dataset. The reason may be that KNRM utilize term-level fine-grained interaction between query and document, which is more beneficial than using contextual information on this test set.

4.4 Ablation Analysis

To evaluate the effectiveness of the three major parts in our model, including query term weighting, representation-based matching and term-based interaction, we perform ablation analysis on the AOL dataset and Tiangong-ST Manual dataset. Specifically, we remove one component at a time for comparison in the following.

- **w/o. QCC**. We remove the Query Change Classifier and train the model with the single task of document ranking.
- **w/o. semantic**. We remove the representation-based matching part, which means we only perform term-level interaction between candidate and session context, without considering to model the query sequence for overall user intent understanding.
- **w/o. removed**, **w/o. added** and **w/o. retained**. We remove three types of fine-grained interactions in the term-based interaction part respectively.

The experimental results are shown in Table 3, and we can get the following conclusions through the ablation analysis:

(1) **The introduction of query change classification task can improve the model's performance**. When removing the query change classification component in HQCN, the performance of the model on all evaluation metrics declines. However, HQCN w/o. QCC still outperforms all the baselines, which indicates that the introduction of this auxiliary task is not the most important reason for the improvement of HQCN. It mainly improves the learning effect of term weights through weakly supervised learning.

Table 3: Results of ablation analysis on the two datasets

Dataset	**AOL**		**TG-ST Manual**	
NDCG	@5	@10	@5	@10
w/o. retained	0.5352	0.5654	0.7716	0.8897
w/o. added	0.5368	0.5678	0.7618	0.8932
w/o. removed	0.5434	0.5724	0.7644	0.8914
w/o. semantic	0.5420	0.5746	0.7722	0.8951
w/o. QCC	0.5520	0.5834	0.7740	0.8953
HQCN	**0.5783**	**0.6070**	**0.7783**	**0.8976**

(2) **It's helpful to model semantic level query change through representation based matching**. When we remove the representation based matching component in the model, the effectiveness of HQCN decreases on both data sets. By analyzing the model structure, it is intuitive that if the representation-based matching part is removed, we can only model the relationship between each pair of adjacent queries, but cannot model the relationship between all queries in the entire session, which makes it difficult to understand the overall search intent of the user.

(3) **Term-based interaction between the candidate document and historical queries and clicks is effective, and the importance of different types of term weights are slightly different**. When we remove the term-based interaction by three types of term weights respectively, the performances of resulted models are all worse than HQCN. Additionally, these results are also worse than that of HQCN w/o. semantic. This indicates that term-based interaction is more effective since it performs fine-grained interaction between all the terms in the candidate document and historical queries. Furthermore, we find the removed term weights are less important than the other two types of term weights, i.e.,

Table 4: Effectiveness of modeling historical query change. TG-MAN is short for Tiangong-ST Manual annotated set.

Model	Dataset	NDCG@5		NDCG@10	
CQCN	AOL	0.5558	-3.88%	0.5868	-3.32%
HQCN	AOL	0.5783	-	0.6070	-
CQCN	TG-MAN	0.7701	-1.05%	0.8938	-0.42%
HQCN	TG-MAN	0.7783	-	0.8976	-

HQCN w/o. removed outperforms HQCN w/o. retained and HQCN w/o. added. One possible reason is that the removed words in the historical query may not appear in any candidate document, so the contribution to the document ranking is limited.

4.5 Effectiveness of Query Change Modeling

One important contribution of HQCN is proposing to model change between adjacent historical queries. In this part, We design a comparison model, named CQCN, to confirm the effectiveness of adjacent historical query change modeling.

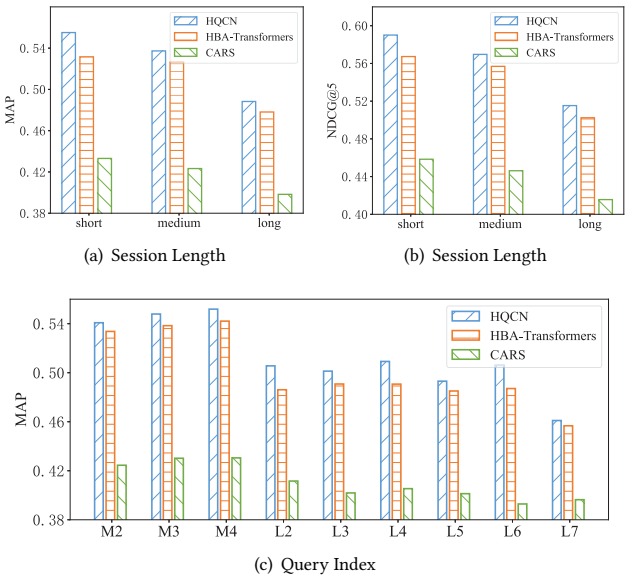
In HQCN, we obtain the term weights by comparing the query with its adjacent queries and clicks. As a comparison, we calculate these term weights by making direct interaction between each historical query with the current query. Specifically, when calculating the retained term weights, we change the $\boldsymbol{h}_l$ in Equation(3) to $\boldsymbol{r}^{\mathrm{q}}_{t,l}$, which means we directly compare the query q_i with the current query q_t. Then we utilize the same way to obtain the added term weights and the removed term weights in CQCN. The meaning of the added term weights and the removed term weights in CQCN are the same, but to keep the structure of HQCN and CQCN consistent, we use different parameters to calculate these two types of term weights. The experimental results are shown in Table 4.

According to the experimental results, if we separately compare each historical query with the current query to calculate the term weights, the model effect will decrease. Because this approach ignores the change between adjacent historical queries, which can describe the change process of user search intent and discover more information omitted by the user in the current query.

4.6 Impact of Session Length and Query Index

In this section, we discuss the impact of session length and query index on model performance. We divide the session into short session (length <= 2), medium session (length= 3 or 4) and long session (otherwise) three categories, and compare MAP and NDCG@5 of HQRN, HBA-Transformers and CARS on them. The experimental results on AOL search logs are shown in Fig. 3 (a) and (b). According to the results, the performance of the three models will decrease as the session length increases. The reason is that long sessions are usually more difficult, so the ranking performance will be worse.

In addition, we analyze model performance on different query indexes. We calculate the MAP of CARS, HBA-Transformers, and HQCN for queries in the medium sessions and long sessions. The results on AOL search logs are shown in Fig. 3 (c). According to the results, all the models perform better as the search task progresses in the medium sessions. Specifically, HQCN increases by 1.31% from M2 to M3, and increases 0.73% from M3 to M4. The reason



(a) Session Length (b) Session Length

(c) Query Index

Figure 3: Model performance on different session lengths and different query indexes on AOL search logs.

is that more information is available as the search task progresses. However, in long sessions, the performance of the models don't improve with the progress of the search task. One reason is that there are not many long sessions (average session length on AOL test set is 2.593), the statistical results are not stable enough. The other reason is that the proportion of noise information in the context of a long session will increase, and the noise information will affect the document ranking performance of current query.

5 CONCLUSION

In this paper, we propose the HQCN to improve session search by modeling multi-granularity historical query change. To model term level query change, we design the QCMU to calculate three types of term weights and utilize them to perform term-based interaction between candidate document and session context. Furthermore, we introduce the query change classification task as an auxiliary task to improve term weight learning. To model semantic level query change, we adopt Transformers [28] to model the query sequence and perform representation-based matching between the candidate document and the session context. Experiments on AOL and Tiangong-ST two search logs confirm the effectiveness of HQCN.

ACKNOWLEDGMENTS

Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 61872370 and No. 61832017, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, China Unicom Innovation Ecological Cooperation Plan, and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. We also wish to acknowledge the support provided and contribution made by Public Policy and Decision-making Research Lab of RUC.

REFERENCES

[1] Eugene Agichtein, Ryen W. White, Susan T. Dumais, and Paul N. Bennett. 2012. Search, interrupted: understanding and predicting search task continuation. In *SIGIR*. ACM, 315–324.

[2] 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, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJ1nzBeA-

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

[4] Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li. 2009. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl (Eds.). ACM, 191–200. https://doi.org/10.1145/1526709.1526736

[5] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. TianGong-ST: A New Dataset with Large-scale Refined Real-world Web Search Sessions. In *Proceedings of the 28th ACM International on Conference on Information and Knowledge Management*. ACM.

[6] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. In *CIKM*. ACM, 1747–1756.

[7] Dongyi Guan, Sicong Zhang, and Hui Yang. 2013. Utilizing query change for session search. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai (Eds.). ACM, 453–462. https://doi.org/10.1145/2484028.2484055

[8] Christophe Van Gysel, Evangelos Kanoulas, and Maarten de Rijke. 2016. Lexical Query Modeling in Session Search. In *ICTIR*. ACM, 69–72.

[9] Kishaloy Halder, Heng-Tze Cheng, Ellie Ka In Chio, Georgios Roumpos, Tao Wu, and Ritesh Agarwal. 2020. Modeling Information Need of Users in Search Sessions. *CoRR* abs/2001.00861 (2020).

[10] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to Rewrite Queries. In *CIKM*. ACM, 1443–1452.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[12] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*. 2042–2050.

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi (Eds.). ACM, 2333–2338. https://doi.org/10.1145/2505515.2505665

[14] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. 2009. Patterns of query reformulation during Web searching. *J. Assoc. Inf. Sci. Technol.* 60, 7 (2009), 1358–1371.

[15] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *SIGIR*. ACM, 445–454.

[16] 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*, Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (Eds.). ACM, 197–206. https://doi.org/10.1145/3269206.3271808

[17] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, Mirella Lapata, Phil Blunsom, and Alexander Koller (Eds.). Association for Computational Linguistics, 427–431. https://doi.org/10.18653/v1/e17-2068

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[19] Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win search: dual-agent stochastic game in session search. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin (Eds.). ACM, 587–596. https://doi.org/10.1145/2600428.2609629

[20] Bhaskar Mitra. 2015. Exploring Session Context using Distributed Representations of Queries and Reformulations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto (Eds.). ACM, 3–12. https://doi.org/10.1145/2766462.2767702

[21] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 1291–1299. https://doi.org/10.1145/3038912.3052579

[22] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Infoscale (ACM International Conference Proceeding Series, Vol. 152)*. ACM, 1.

[23] Chen Qu, Chenyan Xiong, Yizhe Zhang, Corby Rosset, W. Bruce Croft, and Paul Bennett. 2020. Contextual Re-Ranking with Behavior Aware Transformers. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1589–1592. https://doi.org/10.1145/3397271.3401276

[24] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389. https://doi.org/10.1561/1500000019

[25] Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2005. Context-sensitive information retrieval using implicit feedback. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait (Eds.). ACM, 43–50. https://doi.org/10.1145/1076034.1076045

[26] Marc Sloan, Hui Yang, and Jun Wang. 2015. A term-based methodology for query reformulation understanding. *Inf. Retr. J.* 18, 2 (2015), 145–165.

[27] 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*, James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu (Eds.). ACM, 553–562. https://doi.org/10.1145/2806416.2806493

[28] 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: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[29] Gaurav Verma, Vishwa Vinay, Sahil Bansal, Shashank Oberoi, Makkunda Sharma, and Prakhar Gupta. 2020. Using Image Captions and Multitask Learning for Recommending Query Reformulations. In *ECIR (1) (Lecture Notes in Computer Science, Vol. 12035)*. Springer, 681–696.

[30] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A Theoretical Analysis of NDCG Type Ranking Measures. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA (JMLR Workshop and Conference Proceedings, Vol. 30)*, Shai Shalev-Shwartz and Ingo Steinwart (Eds.). JMLR.org, 25–54. http://proceedings.mlr.press/v30/Wang13.html

[31] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query Suggestion with Feedback Memory Network. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 1563–1571. https://doi.org/10.1145/3178876.3186068

[32] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. 2010. Context-aware ranking in web search. In *SIGIR*. ACM, 451–458.

[33] 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, Shinjuku, Tokyo, Japan, August 7-11, 2017*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 55–64. https://doi.org/10.1145/3077136.3080809

[34] Sicong Zhang, Dongyi Guan, and Hui Yang. 2013. Query change as relevance feedback in session search. In *SIGIR*. ACM, 821–824.