A Cooperative Neural Information Retrieval Pipeline with Knowledge Enhanced Automatic Query Reformulation

Xiangsheng Li[1], Jiaxin Mao[2], Weizhi Ma[3], Zhijing Wu[1], Yiqun Liu[1*], Min Zhang[1], Shaoping Ma[1], Zhaowei Wang[4] and Xiuqiang He[4]
[1]Department of Computer Science and Technology, Institute for Artificial Intelligence, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China
[3]Institute for AI Industry Research (AIR), Tsinghua University, Beijing 100084, China
[4]Noah's Ark Lab, Huawei
yiqunliu@tsinghua.edu.cn

ABSTRACT

This paper presents a neural information retrieval pipeline that integrates cooperative learning of query reformulation and neural retrieval models. Our pipeline first exploits an automatic query reformulator to reformulate the user-issued query and then submits the reformulated query to the neural retrieval model. We simultaneously optimize the quality of reformulated queries and ranking performance with an alternate training strategy where query reformulator and neural retrieval model learn from the feedback of each other. Besides, we incorporate knowledge information into automatic query reformulation. The reformulated queries are further improved and contribute to a better ranking performance of the following neural retrieval model. We study two representative neural retrieval models KNRM and BERT in our pipeline. Experiments on two datasets show that our pipeline consistently improves the retrieval performance of the original neural retrieval models while only increases negligible time on automatic query reformulation.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Neural IR; Query reformulation; Knowledge graph

ACM Reference Format:
Xiangsheng Li, Jiaxin Mao, Weizhi Ma, Yiqun Liu, Min Zhang, Shaoping Ma, Zhaowei Wang and Xiuqiang He. 2022. A Cooperative Neural Information Retrieval Pipeline with Knowledge Enhanced Automatic Query Reformulation. 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/10.1145/3488560.3498516

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

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498516

1 INTRODUCTION

Neural retrieval models learn distributed representations of query and documents and conduct semantic matching in the embedding space [31]. A bottleneck of information retrieval is that the short queries issued by users may vaguely represent their true information needs [21]. According to the investigations conducted on large-scale commercial search engines, a query often contains less than three terms and over 16% of queries are ambiguous [17]. Therefore, to accurately understand users' search intents behind these queries is challenging for search engines. To a certain extent, these queries limit the upper bound of ranking performance in existing neural retrieval models.

To address this problem, automatic query reformulation technique [22] is proposed to alter a given query to better represent user's search intent and improve retrieval performance. The basic idea of these methods is to extract terms from top retrieved documents (which is called pseudo-relevance feedback (PRF)) and select terms to expand the original query. Automatic query reformulation has been demonstrated effective and necessary in information retrieval in many applications [6, 22, 33]. However, these studies consider query reformation as an independent learning task before ranking task and do not consider its interactions with the following ranking models. This motivates us to build an integral framework with query reformulation and ranking models.

In this paper, we design a new Cooperative Neural Information Retrieval pipeline (CNIR) with automatic query reformulation. The framework is shown in Figure 2. Different from the traditional neural IR framework, we incorporate a parameterized query reformulation module before neural retrieval model. User-issued query is first reformulated with an automatic query reformulator and then passed to the following neural retrieval model. Inside the pipeline, automatic query reformulator and neural retrieval models aim to optimizing the quality of reformulated queries and ranking performance, respectively. Two modules learn from the feedback of each other to improve the ranking performance, where query reformulator provides the queries with better quality to neural retrieval

*Corresponding author.

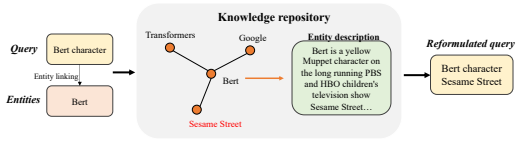

Figure 1: An example of using knowledge information for query reformulation.

models while the neural retrieval model returns the ranking results and the corresponding ranking metric to query reformulator.

The automatic query reformulator is based on reinforcement learning. The *action* is to sample K terms to expand the original query while the *reward* is the ranking performance metrics of the ranked lists retrieved by the original query. Therefore, the query reformulator can generate better queries by learning to maximize the reward through training. For neural retrieval model, it takes as input the reformulated query instead of the original query. The reformulated query with better quality is assumed to better represent users' information needs and thus to achieve a better ranking performance than the original query. We exploit an alternate training strategy to jointly optimize the query reformulator and the neural retrieval model in our pipeline, i,e., the parameters of one module is fixed while training the other. Two modules are prompted alternately to reformulate better queries and improve retrieval performance.

To further improve the quality of the reformulated query, we incorporate knowledge information into the automatic query reformulation. Previous works on automatic query reformulation primarily expand the query using pseudo-relevance feedback (PRF) (e.g., top retrieved documents) [33]. However, PRF is often invalid when the original query is misinterpreted by the search systems and the top retrieved documents are not relevant to the query. In general, knowledge offers two types of information: the relationships between entities, which are often represented as a (knowledge) graph, and the descriptions of entities. Both types of knowledge are useful for automatic query reformulation. For example, in Figure 1, a user issues a query "Bert character" to find a muppet character. Retrieved documents from PRF are mostly about the pretrained language model, which do not match the user's search intent. Knowledge information provides accurate information about the television show and thus can be used to reformulate a better query.

To verify the flexibility and effectiveness of the proposed framework, we test the proposed pipeline with two neural retrieval models in this study, including a kernel based neural ranking model, KNRM [31] and a pretrained contextualized ranking model (BERT) [8]. We conduct experiments on a large-scale public test collection [4] Tiangong-ST and a released testing set from Sogou-QCL [39]. Experimental results show that our pipeline can effectively improve the retrieval performance of neural retrieval models. Meanwhile, we find using knowledge information to build automatic query reformulation module can further improve the ranking performance of our pipeline. Finally, we show that the time overheads introduced by our framework is acceptable for practical retrieval systems.

The primary contributions of our work are three-folds:

(1) We present a Cooperative Neural Information Retrieval pipeline (CNIR) with automatic query reformulation, where query reformulation and neural retrieval model in the pipeline are optimized alternately from the feedback of each other.
(2) We incorporate knowledge information to build a knowledge enhanced query reformulation module, which further improve the ranking performance of the proposed pipeline.
(3) Extensive experiments on two public test collections show that our pipeline effectively improves the retrieval performance of two representative neural retrieval models and the pipeline is also computationally acceptable for practical retrieval systems.

2 RELATED WORK

Query reformulation is categorized into query expansion (or query anchoring) and query rewriting. The former aims to select and add terms to the original query while the latter is detailed rephrase of the original query with seq-to-seq model [13]. In our work, we focus on query expansion with the goal of minimizing query-document mismatch. The term mismatch problem is mainly caused by polysemy (same words with different meanings, e.g., Apple) and synonymy (different words with the same or similar meanings, e.g., USA and America). Early work investigated a range of seminal techniques such as vector feedback [25], co-occurrent terms [11] and comparative analysis of term distributions [9]. To enlarge the vocabulary correlated with the query terms, a variety of data sources are employed to improve the quality of reformulated queries, such as synonyms dictionary [20] and top retrieved documents [33]. The latter is also called pseudo relevance feedback (PRF), which is popularly exploited in different fields to improve the query representation [34, 36]. Some studies [34] only simply take all the PRF terms as side information to improve the query representation. To select terms with better quality, different strategies to estimate the relevance of PRF terms were proposed, such as using probability language model [6], supervised classification methods [3] and reinforcement learning technique [22]. However, these methods are mostly built on a unsupervised ranking model (e.g., BM25). The cooperative optimization between automatic query reformulation and neural retrieval models remain to be investigated.

On the other hand, short and ambiguous queries are the bottleneck of information retrieval, which are often vague to represent users' information needs [21] and limit the ranking performance of most neural retrieval models. Previous works on few shot retrieval [35] and weak supervision [38] enlarge the training corpus and learn a more robust retrieval model. Different from these methods, our pipeline directly learns from the original corpus and does not incorporate weak supervised signals into the training process.

3 PIPELINE FRAMEWORK

3.1 Overview

The overall framework of our proposed Cooperative Neural Information Retrieval (CNIR) pipeline is shown in Figure 2. The pipeline is based on the assumption that queries with better quality can better represent users' information needs and thus achieve a better ranking performance than the original query [7]. Compared with the traditional neural IR pipeline, our pipeline first reformulates the original query by a query reformulator and then takes the reformulated query as inputs into neural retrieval model. The ranking results of the reformulated queries are passed back to the query reformulator and yield ranking metrics as a feedback signal to reflect the quality of the reformulated query. For the neural retrieval

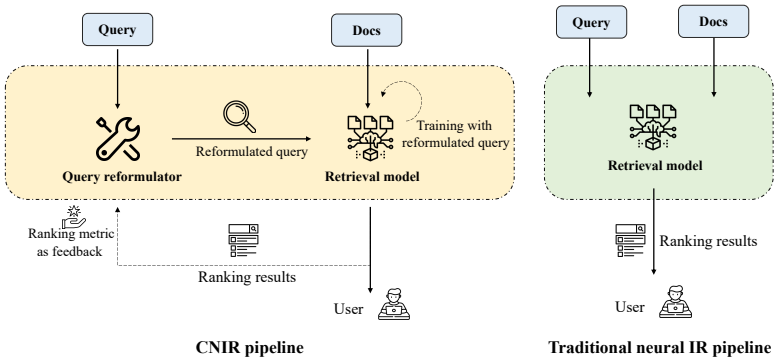

Figure 2: The framework of the proposed Cooperative Neural Information Retrieval pipeline (CNIR). Query reformulator is exploited in both training and inference stage in CNIR pipeline.

model, it uses the reformulated query instead of the original query as inputs to optimize the ranking performance. Two modules in the pipeline are alternately optimized with the feedback of each other. Details of two modules are described as follows.

3.2 Automatic Query Reformulation

Automatic query reformulation module is based on reinforcement learning (RL), which can be divided into two parts: 1) candidate terms construction; 2) automatic query reformulation with RL.

3.2.1 Candidate terms construction. Besides the top retrieved documents (i.e., PRF), candidate terms for automatic query reformulation are also from an external knowledge repository. We construct an entity linker to find knowledge information contained in the issued query. To ensure efficiency, we apply an off-the-shelf entity linker to extract the entities in the query, namely the commonness (popularity) based entity linker [12]. This linker identifies the query entities that are contained in a given large-scale knowledge repository. According to the statistics of our experimental dataset in Figure 4, over 95.9% of queries contain at least one entity in our dataset. We then retrieve the neighbor entities that are connected to the query entities in the knowledge graph. The neighborhoods reveal the entities in the relationship of equivalency and derivation with the query entities, e.g., *the United States-USA* and *Los Angles-USA*, respectively.

To remove the noisy candidate terms, we further filter out the noisy terms according to their word embedding similarity with the query terms. The top knowledge based candidate terms C_{know} with highest similarities are kept to combine with PRF terms C_{PRF} as the final knowledge enhanced candidate terms for the original query q, i.e., $C_q = C_{know} \cup C_{PRF}$. If there is no entity found in the query, the candidate terms only consist of the PRF terms as the previous work [22], i.e., $C_q = C_{PRF}$.

3.2.2 Automatic query reformulation with RL. Given the knowledge enhanced candidate set C_q, we aim to strategically select useful terms for automatic query reformulation. The expanded query is required to boost the retrieval performance compared to the original query. We thus exploit a reinforcement learning strategy to reformulate queries such that the retrieval performance under the reformulated queries is improved. The architecture of our automatic query reformulator is shown in Figure 3. We then describe the *State*, *Action*, *Reward*, and the learning via *Policy Gradient*.

State representation: The state represents whether a candidate term is suitable for the given query. Specifically, for a given query $\boldsymbol{q} = [w_1, w_2, \ldots, w_n]$, we extract the entities that are contained in this query, denoting as $[e_1, e_2, \ldots, e_m]$. The input sequence to the reformulator agent is denoted as the concatenation of the word and entity embedding $\boldsymbol{q'} = \{w_1, w_2, \ldots, w_n, e_1, e_2, \ldots, e_m\}$. The convolutional layer of the agent is defined as:

$$conv_i(z, \mathcal{K}_i) = MaxPool(ReLu(\phi(z))) \tag{1}$$

where z is the input features, ϕ is the convolutional operation; *MaxPool* is the max-pooling operation and $\mathcal{K}_i$ is the kernels to be learned in the ith convolutional layer.

The query representation is then computed by multi-layer convolutional neural network, as follows:

$$z_l = conv_i(z_{l-1}, \mathcal{K}_l), l = 1, \ldots, N \tag{2}$$

where $z_0 = \boldsymbol{q'}$ and the final query representation is $\hat{q} = z_N$.

For each candidate term, we employ multi-layer neural networks to obtain a new vector representation $c'_j = MLP(c_j)$. The state representation is thus the concatenation between query representation and candidate vector representation:

$$s_j = [\boldsymbol{q'} \circ c'_j] \tag{3}$$

Action: The action is to sample K candidate terms based on their state representation and append the sampled terms to the original query to form a new query. The candidate terms are from the combination of PRF terms and knowledge based candidate terms in Section 3.2.1, i.e., C_q. The action probability $\pi(s_j) = P(c_j|\boldsymbol{q})$ of each candidate term is then estimated by:

$$P(c_j|\boldsymbol{q}) = softmax(U^T tanh(W[\boldsymbol{q'} \circ c'_j]) + b) \tag{4}$$

where U, W are the weights and b is the bias. K is the hyper-parameter in our experiment. Note each term is independently sampled by Equation 4 and the reward is obtained by inputting the newly reformulated query to the retrieval model.

Reward: The goal of the query reformulator is to improve the ranking performance of the retrieved documents from the original queries, thus reward function can be any retrieval metrics. In our work, we use mean average precision (MAP) as the reward function.

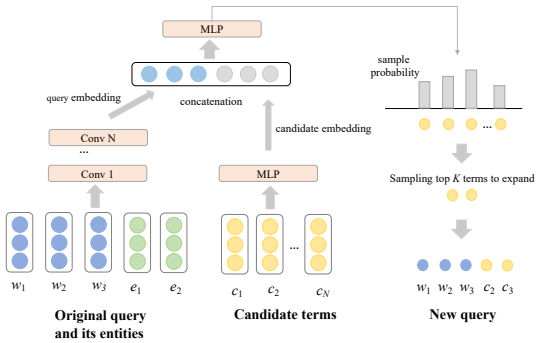

Figure 3: Architecture of the automatic query reformulator.

Policy gradient: We train the query reformulator by REINFORCE algorithm [29]. The gradient of the policy is given by

$$\begin{aligned}\nabla \mathcal{J}_\Theta(\Theta) &= \mathbb{E}_{\pi_\Theta}\left[\sum_{k=1}^{K} \nabla \log \pi(a_k|\Theta) \cdot R\right] \\ &\approx \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{K} \nabla \log \pi(a_k|\Theta) \cdot R_m\end{aligned} \tag{5}$$

where Θ denotes the parameters of the query reformulator and K is the number of expanded terms. M is the sampled number. The expectation reward guides the updates of the parameters Θ of the agent policy $\pi(s_j)$.

3.3 Neural Retrieval Model Training

We take an assumption to train neural retrieval model in our pipeline: queries with better quality can better represent users' information needs and thus achieve a better ranking performance than the original query [7]. Different from the previous work [26], the retrieval model only processes the reformulated query. It outputs the ranking results based on the retrieved documents from the original query. In our work, we study two representative neural retrieval models, i.e., a kernel based neural ranking model (KNRM) [31] and a pretrained contextualized ranking model (BERT) [8].

3.3.1 Kernel based Neural Ranking Model (KNRM). KNRM [31] is a neural retrieval model that uses kernel pooling to model multi-level semantic similarity signals. Given word embeddings of a query and a document, it first constructs an interaction matrix M to model word-level similarities, where each element in M is the cosine similarity between a query word w_i and a document word w_j:

$$M_{ij} = cos(\boldsymbol{v}^q_{w_i}, \boldsymbol{v}^d_{w_j}) \tag{6}$$

where $\boldsymbol{v}_w$ is the word embedding of w. *Kernel pooling* technique is then applied to map the interaction matrix into soft-matched ranking features. In particular, KNRM uses T Gaussian kernels to count the soft matches of interaction pairs at T different strength levels. Each kernel summarizes the interaction features in matrix M as soft similarity counts in the region defined by its mean μ_t and width σ_t, generating a T-dimensional feature vector $\phi(M) = \{K_1(M), \ldots, K_T(M)\}$:

$$\begin{aligned}K_t(M) &= \sum_i \log K_t(M_i) \\ K_t(M_i) &= \sum_j \exp(-\frac{(M_{ij}-\mu_k)^2}{2\sigma_t^2})\end{aligned} \tag{7}$$

Algorithm 1: Cooperative training of the CNIR pipeline.

Input: Training collection $\mathcal{S} = \{(q_1, \boldsymbol{D}_1), \ldots, (q_T, \boldsymbol{D}_T)\}$, where $\boldsymbol{D}_i = \{d_1, \ldots, d_L\}$ is the document list of q_i, query reformulator (QR) and neural retrieval model (NRM).
Output: The ranking results of each $\boldsymbol{D}_i$.

Pretrain NRM with the training collection by Equation 11.
while *convergence condition is not attained* **do**
  Freeze the parameters of NRM and unfreeze QR.
  for *each sample* $(q_i, \boldsymbol{D}_i)$ *in* $\mathcal{S}$ **do**
    QR reformulates q_i to $q_i^{'}$ by Equation 4.
    Compute the ranking metric R_i based on the ranking results of $q_i^{'}$.
    Train QR by Equation 5.
  if *epoch % TRAIN_RANKER_FRE == 0* **then**
    Freeze the parameters of QR and unfreeze NRM.
    for *each sample* $(q_i, \boldsymbol{D}_i)$ *in* $\mathcal{S}$ **do**
      QR reformulates q_i to $q_i^{'}$ by Equation 4.
      Train NRM with the sample $(q_i^{'}, \boldsymbol{D}_i)$ by Equation 11.
  epoch = *epoch* + 1.

The final relevance score is computed with a standard ranking layer:

$$f(\boldsymbol{q}, \boldsymbol{d}) = tanh(w_r \cdot \phi(M) + b_r) \tag{8}$$

with parameters w_r and b_r and $tanh$ activation layer. Compared with BM25, KNRM captures semantic relevance signals and obtains better retrieval performance [30].

3.3.2 BERT. BERT [8] is a pretrained contextualized transformer based language model, which has shown state-of-the-art retrieval performance in many IR-related tasks [23]. BERT concatenates the query and document into a text sequence and feeds it into a multi-layer pretrained transformer architecture:

$$BERT(\boldsymbol{q}, \boldsymbol{d}) = Transformers([CLS] \circ \boldsymbol{q} \circ [SEP] \circ \boldsymbol{d} \circ [SEP]) \tag{9}$$

The special token representation $[CLS]$ in the last layer is considered as the matching feature between query $\boldsymbol{q}$ and document $\boldsymbol{d}$. The ranking layer is applied to the matching feature to estimate the final relevance score:

$$f(\boldsymbol{q}, \boldsymbol{d}) = tanh(w_r \cdot BERT_{CLS}(\boldsymbol{q}, \boldsymbol{d}) + b_r) \tag{10}$$

where w_r and b_r are the learning-to-rank parameters.

3.4 Cooperative Training Procedure

Note that we do not retrieve documents again by the reformulated queries. Instead, we aim to optimize the ranking performance of the retrieved documents from the original queries. Automatic query reformulation module and neural retrieval model are learned based on the feedback of each other. We propose an alternate training strategy to train our pipeline, as detailed in Algorithm 1. We initialize the neural retrieval model by training on the original training collection. For the training process of CNIR, the query reformulator is trained first with the reward obtained from the neural retrieval model. It encourages the reformulator to reformulate queries with

Table 1: Statistics of the datasets. Test_1 and Test_2 indicate the testing set of Tiangong-ST and Sogou-QCL, respectively.

	Train	Valid	Test_1	Test_2
#queries	344,942	4,888	2,000	900
#sessions	143,155	2,000	2,000	na.
#avg. doc per query	9.60	9.58	9.59	17.78

better quality. Then, the reformulated queries are used to fine-tune of the neural retrieval model, which facilitates the retrieval model to better estimate the document relevance.

Two modules in CNIR pipeline are alternately optimized until the ranking performance converges. Specifically, we use the pairwise learning-to-rank loss to train the neural retrieval models during both the pretraining and fine-tuning stage:

$$\mathcal{L}_{rank} = \max(0, 1 - f(q, d^+) + f(q, d^-)) \tag{11}$$

where d^+ is a document that is more relevant to the query q than the document d^-.

4 EXPERIMENTAL SETUP

4.1 Dataset

To evaluate the performance of our framework, we conduct experiments on a large-scale public available benchmark data (Tiangong-ST [2] [4]) and a released testing set from Sogou-QCL [3] [39]. Table 1 shows the statistics of the datasets. Tiangong-ST provides web search session data from an 18-day search log. It contains weak relevance labels (i.e., click relevance labels [31]) derived by six different click models for all query-document pairs and human relevance labels for documents in the last query of 2,000 sampled sessions. To ensure efficiency when training the query reformulator, we exploit document titles in both training and testing instead of the full document content. Similar experimental setups have been used in previous studies (e.g., [18]). Prior studies [5, 31] have shown that weak relevance labels derived from click models can be used to train and evaluate retrieval models. Since the Partially Sequential Click Model (PSCM) [27] achieves the best relevance estimation performance among the six click model alternatives, we employ click labels from the PSCM for training and validation. Training on weak relevance labels also enables our pipeline to obtain an immediate reward about the ranking results.

We evaluate our framework on two datasets: Tiangong-ST testing set and Sogou-QCL testing set. Tiangong-ST testing set is built from the last query of 2,000 sampled sessions. We use click relevance labels from the same PSCM and the provided five-graded human annotated relevance labels for evaluation. In Sogou-QCL testing set, we used click relevance labels from PSCM for evaluation.

Entity annotation: We utilize XLore [28] as our knowledge graph foundation. XLore is an English-Chinese bilingual knowledge graph built from English Wikipedia, Chinese Wikipedia, Baidu Baike and Hudong Baike. It contains 16,284,901 entities, 2,466,956 concepts and 446,236 relations. The relations have four types: *subclass*, *instanceof* and *same*, *related*, where the proportions are 2.6%, 42.1%, 7.9%, 47.3%, respectively. The relations *subclass* and *instanceof* are further categorized as Derivation type while *same* and *related* are considered to be Equivalency type. These relations discover more relevant items compared with PRF and thus provide potential to improve query reformulation. The distribution of the entity number in the query in the training set is shown in Figure 4, where over 95.9% of queries have at least one entity and the average entity number in the query is 3.44.

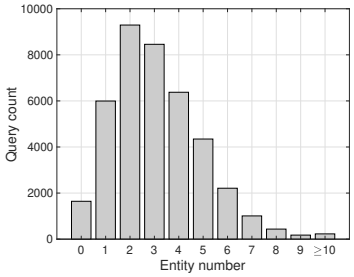

Figure 4: The distribution of entity number in the query.

Candidate terms: The candidate terms come from two sources: pseudo relevance feedback and knowledge information. We use the terms from top 3 documents retrieved by BM25 as the pseudo relevance feedback terms (17.2 terms on average). Knowledge terms are first filtered if they are duplicated with PRF terms. Specifically, we keep the top 20 terms according to the embedding similarity with the query terms as the knowledge enhanced candidate terms. The number of expanded terms K is set as 3 [4].

4.2 Experimental Settings

Since our pipeline consists of automatic query reformulation and neural retrieval model, we compare our pipeline with existing neural retrieval models and study the effectiveness of different query reformulation methods in our pipeline .

4.2.1 Retrieval Model Baselines.

- **BM25**: A popular probabilistic bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document [24].
- **ARCII** [14]: A neural retrieval model that maps the word embeddings of query and document to an aggregated embedding by a CNN. we use a two-layer CNN, where the size of kernels and pooling in both layers are set to (3×3) and (2×2). There are 16/32 kernels in two layers.
- **ARCI** [14]: ARCI is a representation-based model which encodes text information by CNNs. We use a three-layer CNN where the filter windows sizes are 1 to 3 and there are 64 feature maps for each filter.
- **DSSM** [16]: DSSM is also a representation-based model. It consists of a word hashing layer, two non-linear hidden layers, and an output layer. We use a three-layer DNN as in the original paper of DSSM; the hidden number of each layer is set to 50.
- **KESM** [32]: KESM is knowledge-aware neural retrieval model baseline, which models the salience of query entities in candidate

[3] Pseudo click drawn from PSCM labels, where documents with top 29% click probability (grade 1 or higher) are considered being clicked.
[2] http://www.thuir.cn/tiangong-st/.
[3] http://www.thuir.cn/sogouqcl/.

[4] We find expanding more than 3 terms do not have significant difference on retrieval performance but it increases much training time. Thus expanding 3 terms is a suitable setting in our work.

Table 2: **Retrieval performance of our CNIR pipeline with existing retrieval models. The subscript P and K denote the PRF and Knowledge enhanced candidate term set, respectively. †, 1 and 2 indicate statistically significant improvements over the strongest baseline BERT, the original ranker in CNIR (i.e., KNRM or BERT) and the corresponding CNIR$_P$. (p-value ≤ 0.05)**

	PSCM (Tiangong-ST)				**HUMAN** (Tiangong-ST)				**PSCM** (Sogou-QCL)			
	MAP	**ERR**	**ndcg@5**	**ndcg@10**	**MAP**	**ERR**	**ndcg@5**	**ndcg@10**	**MAP**	**ERR**	**ndcg@5**	**ndcg@10**
BM25	0.4577	0.2928	0.4474	0.6207	0.7743	0.4390	0.6269	0.7890	0.6064	0.4322	0.4790	0.5949
ARCII	0.6803	0.3650	0.6866	0.7854	0.7819	0.4581	0.6889	0.8285	0.6035	0.4283	0.4821	0.6011
ARCI	0.6929	0.3944	0.7012	0.7946	0.7854	0.4673	0.7043	0.8256	0.6023	0.4345	0.4904	0.6056
DSSM	0.7112	0.4312	0.7143	0.8043	0.7842	0.4701	0.7016	0.8253	0.6056	0.4408	0.4989	0.6135
KESM	0.7294	0.4793	0.7447	0.8459	0.7847	0.4789	0.6989	0.8345	0.6171	0.4526	0.5163	0.6293
EDRM	0.7971	0.4912	0.8106	0.8645	0.7984	0.5032	0.7135	0.8389	0.6224	0.4752	0.5519	0.6422
KNRM	0.7748	0.4948	0.7885	0.8507	0.8051	0.5002	0.7119	0.8374	0.6044	0.4485	0.4960	0.6099
CNIR$_P$ + KNRM	0.7901^{1}	0.5020	0.8098^{1}	0.8621^{1}	0.8071	0.5175^{1}	0.7169	0.8451	0.6245^{1}	0.4708^{1}	0.5415^{1}	0.6382^{1}
CNIR$_K$ + KNRM	0.8135^{1,2}	0.5231^{1,2,†}	0.8334^{1,2}	0.8845^{1,2}	0.8097	0.5257^{1}	0.7212	0.8574^{1}	0.6398^{1,2}	0.4892^{1,2}	0.5741^{1,2}	0.6532^{1,2}
BERT	0.8273	0.5012	0.8422	0.8822	0.8086	0.5267	0.7275	0.8511	0.6549	0.5070	0.5771	0.6722
CNIR$_P$ + BERT	0.8475^{1}	0.5224^{1}	0.8693^{1}	0.9031^{1}	0.8093	0.5277	0.7285	0.8519	0.6727^{1}	0.5277^{1}	0.5886^{1}	0.6947^{1}
CNIR$_K$ + BERT	**0.8735**^{1,2}	**0.5426**^{1,2}	**0.8864**^{1,2}	**0.9073**^{1}	**0.8104**	**0.5383**^{1,2}	**0.7405**^{1,2}	**0.8649**^{1,2}	**0.6860**^{1,2}	**0.5367**^{1,2}	**0.5980**^{1,2}	**0.7063**^{1,2}

documents. We set the dimension of word and entity embedding as 50 and remains other parameter settings as the original paper.

- **EDRM** [19]: EDRM is another knowledge-aware neural retrieval model, which considers both semantic similarity and entity relationship. We set the dimension of word and entity embedding as 50 and remains other parameter settings as the original paper.
- **KNRM** [31] : We use 11 kernels as the default setting in the original paper (10 soft-matching and 1 exact-matching kernels).
- **BERT** [8] : The state-of-the-art neural retrieval model. We use the pretrained 12-layer *bert-base-chinese*[5] model.

4.2.2 Automatic Query Reformulation Baselines.

- **TFIDF**: We extract the terms with top K TFIDF values and add them to the original queries to test the retrieval performance.
- **RM**: RM is the Lavrenko's relevance models [6], which is a popular relevance model for query reformulation. We follow the setting in [37] and set the parameter $\lambda = 0.5$.
- **SL**: Following [3], we implement a recurrent neural network (RNN) as the Supervised Learning baseline (SL). We feed query terms into RNN and take the last output embedding as the query representation. Then it is concatenated with the candidate terms to form the state representation, which is feed into a feed-forward neural network to estimate the probability of this word to be chosen. The term is marked as relevant if $(R' - R) > 0.3$, where R' and R are the retrieval performance of the reformulated query and the original query, respectively.
- **RL-CNN** [22]: The recent Reinforcement Learning (RL) query reformulation method. To fairly compare its performance, the CNN architecture is the same as our query reformulator. When it is applied on knowledge enhanced candidate terms, it is the same as our query reformulator in Section 3.2.2.

4.2.3 Parameter settings. The parameters of the query reformulator are optimized by Adagrad optimizer, with a batch size of 50 and a learning rate of 1e-5. Neural retrieval models are trained with Adam optimizer. The learning rate of KNRM during pretraining and fine-tuning is 1e-3 and 1e-4 while BERT always uses a learning rate of 3e-6. The dimension of the word embeddings, entity embedding are both 50. Word embeddings are pretrained on a Chinese Wikipedia dataset while entity embeddings are pretrained on the given graph structure by node2vec [10][6]. For the CNN layer in the query reformulator, the filter windows sizes are 1 to 3 and there are 50 feature maps for each filter. Early stopping with a patience of 10 epochs is adopted during the training process. For each query, we sample 5 possible reformulated queries, i.e., $M = 5$. For neural retrieval models, KNRM and BERT are updated after the query reformulator learns for 10 and 20 epochs, respectively. The source code is publicly available[7].

5 EVALUATION RESULTS

In this section, we present our experimental results to answer the following research questions:

(RQ1) How does our pipeline perform compared to existing neural retrieval models?
(RQ2) How does our pipeline perform when using other query reformulation methods?
(RQ3) Does cooperative training strategy boost the ranking performance of our pipeline?
(RQ4) How efficient is our pipeline in both training and testing?

5.1 Overall ranking performance

This section aims to answer ***RQ1***. We first compare our pipeline with different retrieval model baselines, which includes BM25, neural IR models and knowledge-aware neural IR models (KESM and EDRM). The results are shown in Table 2.

It is observed that our CNIR pipeline conducted on PRF and Knowledge candidate terms (CNIR$_P$ and CNIR$_K$) can outperform BM25, neural IR models and knowledge-aware neural IR models in most evaluation metrics. This shows the retrieval power of our pipeline across different retrieval models. For the comparison with the original neural retrieval model in CNIR pipeline, we find CNIR can significantly improves the ranking performance of KNRM and BERT in most evaluation metrics. Though, the improvements over

[5]https://github.com/google-research/bert/blob/master/multilingual.md

[6]We do not use the popular TransE [2] to pretrain entity embedding because the number of relation types in XLore is small.

[7]https://github.com/lixsh6/WSDM2022-CNIR.

Table 3: Comparison with other automatic query reformulation methods in CNIR$_K$. † indicate significant improvements over the original ranker (Raw). (p-value ≤ 0.05)

	Tiangong-ST		Sogou-QCL	
	ndcg@5	**ndcg@10**	**ndcg@5**	**ndcg@10**
BM25 as ranker				
Raw	0.4474	0.6207	0.4790	0.5949
TFIDF	0.4435	0.6318	0.4112	0.5305
RM	0.4174	0.6165	0.3892	0.4978
SL	0.4694†	0.6384†	0.5053†	0.6271†
RL-CNN	0.4815†	0.6275	0.5124†	0.6242†
KNRM as ranker				
Raw	0.7885	0.8507	0.4960	0.6099
TFIDF	0.7313	0.7905	0.4760	0.5829
RM	0.7283	0.7823	0.4693	0.5702
SL	0.8105†	0.8671†	0.5312†	0.6254†
CNIR(RL-CNN)	0.8334†	0.8845†	0.5741†	0.6532†
BERT as ranker				
Raw	0.8422	0.8822	0.5771	0.6722
TFIDF	0.8252	0.8626	0.5471	0.6322
RM	0.8193	0.8593	0.5413	0.6315
SL	0.8538†	0.8823	0.5874†	0.6863†
CNIR(RL-CNN)	**0.8864**†	**0.9073**†	**0.5980**†	**0.7063**†

human annotated labels in Tiangong-ST are not all significant, it may be because that our pipeline CNIR is trained with pseudo relevance label PSCM, which is somehow inconsistent with human labels. In general, it illustrates that our pipeline with additional automatic query reformulation can effectively improve the ranking performance of the original neural retrieval model. Query reformulator, with the goal of optimizing the ranking metric of a given document list, can learn to generate a better query that represents users' information needs. Due to the improved query, the neural retrieval model can also estimate the relevance of each document better and improve the ranking performance again. Two modules learn from each other and continuously optimize the ranking performance during training.

We also compare our pipeline when incorporating knowledge information into automatic query reformulation. Note CNIR$_K$ is not only different with CNIR$_P$ from the perspective of candidate term construction, but also the additional entity inputs to the CNN architecture in Equation 1. It is observed that when using knowledge information CNIR$_K$ can outperform CNIR$_P$ significantly in most evaluation metrics on KNRM and BERT. On human annotated labels in Tiangong-ST, CNIR$_K$ is also superior to the original BERT significantly on ERR, ndcg@5,10. It suggests that the performance of automatic query reformulation decides the upper bound of ranking performance in our pipeline. If automatic query reformulation is improved, the ranking performance of the pipeline can be better during the cooperative learning process.

5.2 Analysis on query reformulation

This section aims to answer ***RQ2***. We alter the RL based automatic query reformulation method in our pipeline to compare the effectiveness of other query reformulation methods. In addition to neural retrieval models, we also include BM25 as the retrieval model in our pipeline. The results are shown in Table 3. In the scenario with BM25 ranker, our pipeline deteriorates the traditional query reformulation optimization [3, 22] where retrieval models do not require to be trained. *Raw* indicates that the original retrieval model without automatic query reformulation. In the scenario with neural retrieval models, we compare the rule-based query reformulation strategy *TFIDF*, *RM* and a supervised learning method *SL*. Note our query reformulator has the same structure with RL-CNN, RL-CNN with KNRM or BERT indicates our CNIR pipeline.

We find it is hard to improve the ranking performance when applying our pipeline on BM25, rule-based methods cannot improve the ranking performance of BM25. It suggests that both rule-based methods cannot capture the effective terms in relationship with the original query. Supervised learning and reinforcement learning methods learn to reformulate queries according to the feedback of ranking performance, thus are effective to improve the ranking performance of the original ranker BM25. Since RL-CNN learns with a continuous reward against the discrete signal in SL, RL-CNN can distinguish the useful terms better for query reformulation and thus achieves better ranking performance.

When applying neural retrieval models in our pipeline, we find the ranking performance is consistent to the quality of automatic query reformulation methods. Rule-based query reformulation is not effective in our pipeline while supervised learning and reinforcement learning methods contribute to better ranking performance. This phenomenon suggests that the quality of automatic query reformulation methods decides how well the whole pipeline can achieve. This also motivates us to build a knowledge enhanced query reformulator to further improve the ranking performance.

Table 4: Ranking performance when freezing the parameters of neural retrieval model.

	Tiangong-ST		Sogou-QCL	
	ndcg@5	**ndcg@10**	**ndcg@5**	**ndcg@10**
KNRM as ranker				
CNIR$_P$ (Freeze)	0.7764	0.8412	0.5342	0.6289
CNIR$_P$	0.8089	0.8621	0.5415	0.6382
CNIR$_K$ (Freeze)	0.8123	0.8615	0.5351	0.6270
CNIR$_K$	0.8334	0.8845	0.5741	0.6532
BERT as ranker				
CNIR$_P$ (Freeze)	0.8451	0.8844	0.5711	0.6873
CNIR$_P$	0.8693	0.9031	0.5886	0.6947
CNIR$_K$ (Freeze)	0.8421	0.8852	0.5732	0.6892
CNIR$_K$	**0.8864**	**0.9073**	**0.5980**	**0.7063**

5.3 Analysis on cooperative training

This section aims to answer ***RQ3***. We aim to understand how our pipeline performs when it deteriorates to the traditional automatic query reformulation optimization (i.e., the retrieval models are not required to be trained.) We freeze the parameters of neural retrieval models and only train the query reformulator to validate the necessity of our proposed cooperative training strategy. The results are shown in Table 4.

It is found that regardless of candidate terms construction, if KNRM and BERT are not fine-tuned during the training, they cannot perform well on the reformulated queries. It is because neural retrieval models are not trained with these reformulated queries.

Table 5: Comparison of time consumption over Tiangong-ST testing set when using different retrieval frameworks.

	#params	Training (total time)	Testing (full test set)
BM25	n.a	n.a	1.2s
RL-CNN	5M	3.2h	1.3s
KNRM	5M	n.a	1.3s
$CNIR_K$ + KNRM	10M	9.4h	1.4s
BERT	103M	n.a	49.5s
$CNIR_K$ + BERT	108M	38.3h	52.2s

The relevance estimation of these queries may not be accurate and thus yields a wrong signal to train query reformulator. In this scenario, the learning of query reformulator is limited by the ranking power of current neural retrieval models. Ideally, when the power of the query reformulator achieves a upper bound after certain epochs, it captures the possible reformulated queries that the current retrieval models can do well. Thus, it is necessary to conduct cooperative training strategy to make sure both query reformulator and neural retrieval models can be optimized continuously.

5.4 Time efficiency

This section aims to answer ***RQ*4**. Since our pipeline exploits additional automatic query reformulation module compared to other retrieval models, we aim to understand whether it increases acceptable time overheads. We report the time consumption of different retrieval frameworks, as shown in Table 5. Automatic query reformulation is based on knowledge enhanced candidate terms.

The parameter number of our query reformulator is about 5 million. Neural retrieval model is the key module in our pipeline while the parameters in query reformulator is far less than those in the neural retrieval models. Therefore, neural retrieval models with larger complexity require more training time in our pipeline. We can observe that the query reformulator only increases negligible time on the original rankers during inference. However, our pipeline brings significant improvements to the original neural retrieval models. As reported in Table 2, $CNIR_K$ + KNRM can perform closely to BERT while its efficiency is substantially superior to BERT. This makes our framework computationally applicable in practical ranking systems.

6 LIMITATIONS

Despite we find our CNIR pipeline is effective to boost the ranking performance, we admit some limitations of our work and share some directions for the future work.

Reranking vs first stage retrieval: Our experiments are conducted in a reranking setting rather than first stage retrieval. Previous studies showed automatic query reformulation benefits in improving recall in the first stage retrieval [1], but it is not validated in our experiments. The limited improvements when applying on BM25 in Table 3 may also be because rule-based query reformulation is more effective in the first stage retrieval rather than the reranking scenario. Although we do not have experiments in the first stage retrieval, it does not mean that our CNIR pipeline is not effective in the first stage retrieval. In the future, we aim to incorporate embedding retrieval models [15] to further improve the first stage retrieval performance.

Query anchoring vs query rewriting: As discussed in Section 2, query reformulation is categorized into query anchoring and query rewriting. In our experiments, we only focus on query anchoring. Ideally, a good reformulated query that represents users' information needs clearly should not be limited with the original query terms. Therefore, using query rewriting technique remains to be investigated in our framework. The key challenge is to balance the tradeoff between identifying the useless words or mistake words and misjudging the keywords in the original query.

7 CONCLUSION

This paper presents a cooperative neural information retrieval pipeline (CNIR) with knowledge enhanced automatic query reformulation. Different from traditional neural information retrieval framework, our pipeline sequentially reformulates a query first and then submits the reformulated query to neural retrieval models. Automatic query reformulation, which is designed by reinforcement learning, learns to reformulate a query according to the ranking results from neural retrieval models. The reformulated query can better represent user information needs compared to the original query and thus be used in the training of neural retrieval models to further optimize the ranking performance. In the proposed pipeline, we simultaneously optimize the quality of reformulated queries and ranking performance with an alternate training strategy. Experiments show that the alternate training of both modules in CNIR are effective to improve the ranking performance and our pipeline can outperform existing retrieval model baselines. To further improve the ranking performance, we incorporate knowledge information into automatic query reformulation and yield a better CNIR pipeline with better ranking performance. We find that the quality of automatic query reformulation module decides how well the pipeline can achieve. Further detailed analysis evaluates the effectiveness of different query reformulation strategies and time overheads in both training and testing. We find our pipeline can improve existing retrieval models significantly while only increase negligible inference (or ranking) time, which is computationally applicable for practical ranking systems. Our work is a new framework to build neural retrieval models and provides a better understanding of how to use automatic query reformulation to guide the learning of neural retrieval models.

Despite we show different advantages of our CNIR pipeline, we also point out two limitations in our work, as discussed in Section 6. In the future, we aim to focus on these two challenges and build a better ranking system with the proposed framework.

ACKNOWLEDGEMENTS

This work is supported by the Natural Science Foundation of China (Grant No. 61732008, 61902209, U2001212), Beijing Academy of Artificial Intelligence (BAAI) and Tsinghua University Guoqiang Research Institute. This project is also supported by Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100020098) and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. *Computer Science Department Faculty Publication Series* (2004), 189.

[2] Nicolas Usunier Alberto Garcia-Duran Jason Weston Bordes, Antoine and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[3] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 243–250.

[4] 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, 2485–2488.

[5] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. San Rafael: Morgan and Claypool.

[6] W Bruce Croft, Stephen Cronen-Townsend, and Victor Lavrenko. 2001. Relevance Feedback and Personalization: A Language Modeling Perspective.. In *DELOS*. Citeseer.

[7] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 985–988.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Tamas E Doszkocs. 1978. AID, an associative interactive dictionary for online searching. *Online Review* (1978).

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*.

[11] David J Harper and Cornelis Joost Van Rijsbergen. 1978. An evaluation of feedback in document retrieval using co-occurrence data. *Journal of documentation* (1978).

[12] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. *Entity Linking in Queries: Efficiency vs. Effectiveness*. Springer, Cham.

[13] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1443–1452.

[14] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[15] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2333–2338.

[17] Xiangsheng Li, Yiqun Liu, Xin Li, Cheng Luo, Jian-Yun Nie, Min Zhang, and Shaoping Ma. 2018. Hierarchical Attention Network for Context-Aware Query Suggestion. In *Asia Information Retrieval Symposium*. Springer, 173–186.

[18] Xiangsheng Li, Yiqun Liu, Jiaxin Mao, Zexue He, Min Zhang, and Shaoping Ma. 2018. Understanding Reading Attention Distribution during Relevance Judgement. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 733–742.

[19] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. *arXiv preprint arXiv:1805.07591* (2018).

[20] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[21] Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509* (2017).

[22] Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* (2017).

[23] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).

[24] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*.

[25] Joseph Rocchio. 1971. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing* (1971), 313–323.

[26] Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas, and Maarten de Rijke. 2020. Query resolution for conversational search with limited supervision. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 921–930.

[27] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015. Incorporating non-sequential behavior into click models. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 283–292.

[28] Zhigang Wang, Juanzi Li, Zhichun Wang, Shuangjie Li, Mingyang Li, Dongsheng Zhang, Yao Shi, Yongbin Liu, Peng Zhang, and Jie Tang. 2013. XLore: A Large-Scale English-Chinese Bilingual Knowledge Graph. In *Proceedings of the 12th International Semantic Web Conference*. 121–124.

[29] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[30] Yiqun Liu Jiaxin Mao Weizhi Ma Min Zhang Xiangsheng Li, Maarten de Rijke and Shaoping Ma. 2020. Learning Better Representations for Neural Information Retrieval with Graph Information. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. ACM.

[31] 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 Conferenc*. ACM, 55–64.

[32] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tieyan Liu. 2018. Towards Better Text Understanding and Retrieval through Kernel Entity Salience Modeling. (2018), 575–584.

[33] Jinxi Xu and W Bruce Croft. 2017. Quary expansion using local and global document analysis. In *Acm sigir forum*, Vol. 51. ACM New York, NY, USA, 168–175.

[34] Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, Jun Huang, and Haiqing Chen. 2018. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In *The 41st international acm sigir conference on research & development in information retrieval*. 245–254.

[35] Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. 2021. Few-Shot Conversational Dense Retrieval. *arXiv preprint arXiv:2105.04166* (2021).

[36] Hamed Zamani, Javid Dadashkarimi, Azadeh Shakery, and W Bruce Croft. 2016. Pseudo-relevance feedback based on matrix factorization. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 1483–1492.

[37] Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)* 22, 2 (2004), 179–214.

[38] Kaitao Zhang, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2020. Selective weak supervision for neural information retrieval. In *Proceedings of The Web Conference 2020*. 474–485.

[39] Yukun Zheng, Zhen Fan, Yiqun Liu, Cheng Luo, Min Zhang, and Shaoping Ma. 2018. Sogou-qcl: A new dataset with click relevance label. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1117–1120.