Chapter 6
Query Rewriting

Hui Liu, Dawei Yin, and Jiliang Tang

Abstract It is well known that there is a lexical chasm between web documents and user queries. As a result, even when the queries can fully capture users' information needs, the search engines could not retrieve relevant web documents to match these queries. Query rewriting aims to bridge this gap by rewriting a given query to alternative queries such that the mismatches can be reduced and the relevance performance can be improved. Query rewriting has been extensively studied and recent advances from deep learning have further fostered this research field. In this chapter, we give an overview about the achievements that have been made on query rewriting. In particular, we review representative algorithms with both shallow and deep architectures.

6.1 Introduction

With the advance of technologies, information in the web has been increased exponentially. It had become increasingly hard for online users to find information they are interested in. Modern search engines have been proven to successfully mitigate this information overload problem by retrieving relevant information from massive web documents according to users' information needs (or queries) [3]. However, it is well known that there exists a "lexical chasm" [26] between web documents and user queries. The major reason is that web documents and user queries are created by different sets of users and they may use different vocabularies and distinct language styles. Consequently, even when the queries can perfectly match users' information needs, the search engines may be still unable to locate relevant web documents. For example, users want to find price information about

H. Liu · J. Tang (✉)
Michigan State University, East Lansing, MI, USA
e-mail: liuhui7@msu.edu; tangjili@msu.edu

D. Yin
Baidu Inc., Beijing, China
e-mail: yindawei@outlook.com

Y. Chang, H. Deng (eds.), *Query Understanding for Search Engines*,
The Information Retrieval Series 46, https://doi.org/10.1007/978-3-030-58334-7_6

Tesla using a query "price tesla," while such information is expressed as "how much tesla" in web documents indexed by the search engines. Thus, it is demanding that search engines should intelligently match information needs of their users by understanding the intrinsic intent in queries.

Query rewriting (QRW), which targets to alter a given query to alternative queries that can improve relevance performance by reducing the mismatches, is a critical task in modern search engines and has attracted increasing attention in the last decade [11, 20, 26]. Thus, we have witnessed a rapid development of the query rewriting techniques. At the early stage, methods have been developed to find terms related to these in a given query and then substitute terms in the original queries with these related ones (or substitution-based methods). Then if we treat queries as the source language and web documents as the target language, the query rewriting problem can be naturally considered as a machine translation problem; thus, machine translation techniques have been applied for QRW (or translation-based methods) [26]. Recently, deep learning techniques have been widely applied in information retrieval [21] and natural language processing [33]. There are very recent works applying deep learning in query rewriting that achieve the state-of-the-art performance [17]. Thus, in this survey, we will follow the used techniques to review representative query rewriting methods and the structure of the survey is as follows:

- In Sect. 6.2, we will review representative methods with traditional shallow models including substitution-based methods and translation-based methods.
- In Sect. 6.3, we will review algorithms based on deep learning techniques such as word embedding, seq2seq models, deep learning to rewrite frameworks, and deep reinforcement learning.
- In Sect. 6.4, we will conclude the survey and discuss some promising directions in query rewriting.

6.2 QRW with Shallow Models

In the section, we will review shallow query rewriting algorithms including substitute-based and translation-based methods.

6.2.1 Substitution-Based Methods

Given the original query, substitution-based methods aim to generate rewritten queries by replacing the query as a whole or by substituting constituent phrases [20]. There are two key steps for substitution-based methods: substitution generation and candidate selection. The substitution generation step is to find substitutions in the levels of queries, phrases, or terms for the original query. The substitution

generation step can suggest many rewritten candidates for the original query. Thus the candidate selection step is to select good candidates. Many possible resources can be used to generate query or term substitutions. One type resource is static such as WordNet [12] and Wikipedia [32]. However, these static resources generally do not allow us to generate substitutions for new concepts. It is also challenging to consider contextual information. Thus, resources based on users' search feedback have been widely adapted and we will introduce several representative methods in the following.

In [20], user sessions from search query logs have been used for query rewriting. These sessions have been reported to include 50% reformulations [19]. Query reformulation is that a user reformulates a query to other related queries in a query session by inserting, deleting, substituting, or rephrasing words of the original query [2]. Query reformulation is very similar to the query rewriting task; thus, it is natural to use user session reformulation data.

A pair of successive queries issued by a single user on a single day is referred as a candidate reformulation or a query pair. Then, they aggregate query pairs over users. For phrase substitutions, the authors segment the whole query into phrases using point-wise mutual information and find query pairs that differ by only one segment. This pair of phrases is selected as a candidate phrase pair.

To identify highly related query pairs and phrase pairs, the work makes two hypotheses to evaluate that the probability of term q_2 is the same whether term q_1 is present or not.

$$H_1 : P(q_2|q_1) = p = P(q_2|\neg q_1) \tag{6.1}$$

$$H_2 : P(q_2|q_1) = p_1 \neq p_2 = P(q_2|\neg q_1) \tag{6.2}$$

The log-likelihood ratio score based on the probabilities of the two hypotheses is used to measure the dependence between two terms q_1 and q_2.

$$\text{LLR} = -2 \log \frac{L(H_1)}{L(H_2)} \tag{6.3}$$

The query pairs and phrase pairs with a high LLR score are identified as substitutable pairs because of the statistically significant relevance.

The work extracts a list of features from the queries and uses human judgments and machine learning to train a classifier for high quality query suggestions. Since this method could precompute offline the whole-query substitutions and their scores and the edit distance for phrase similarity evaluation, it only requires look-up substitutions at run-time.

In [30], the authors work on mining search engine log data at the level of terms rather than the level of queries. The user session information is leveraged for query refinement. This method is based on an observation that terms with similar meaning tend to co-occur with the same or similar terms in the queries. The associated terms are discovered from search engine logs to substitute the previous terms or add

new terms to the original query. The term associations are extracted based on the context distribution. A contextual model was proposed by investigating the context similarity of terms in historical queries from log data. Two terms in a pair with similar contexts are used to substitute each other in new query generation. The contextual model is defined based on the maximum likelihood estimation

$$P_C(a|\omega) = \frac{c(a, C(\omega))}{\sum_i c(i, C(\omega))} \tag{6.4}$$

where $C(\omega)$ is the context of a word ω. This model evaluates the likelihood of a word a to appear in the context of a given word ω. The Kullback–Leibler (KL) divergence $D(\cdot||\cdot)$ has been used in the language modeling approach to measure the similarity between two contexts. In [30], the metric of the similarity between the original word ω and the candidate word s is given based on the KL-divergence as follows:

$$t_C(s|\omega) = \frac{\exp(-D[P_C(\cdot|s)||\tilde{P}_C(\cdot|\omega)])}{\sum_s \exp(-D[P_C(\cdot|s)||\tilde{P}_C(\cdot|\omega)])} \tag{6.5}$$

where $\tilde{P}_C(\cdot|\omega)$ is the smoothed contextual model of ω using Dirichlet prior smoothing approach. The position information is introduced in the contextual models.

$$\prod_{j=1, i-j>0}^{k} \tilde{P}_{L_{i-j}}(\omega_{i-j}|s) \times \prod_{j=1, i+j\leq n}^{k} \tilde{P}_{L_{i+j}}(\omega_{i+j}|s) \tag{6.6}$$

where k is the number of adjacent terms to be considered. The impact of a word far away from the word in consideration is insignificant. Mutual information is used to capture the relation between two terms over user sessions inside queries.

$$I(s, \omega) = \sum_{X_s, X_\omega \in \{0,1\}} P(X_s, X_\omega) \log \frac{P(X_s, X_\omega)}{P(X_s)P(X_\omega)} \tag{6.7}$$

where X_s and X_ω are binary variables indicating the presence/absence of term s and term ω in each user session. A normalized version of mutual information is generalized to make the mutual information of different pairs of words comparable. Then, all the candidate queries are sorted according to the probability given by Eq. 6.6. The top ranked candidate queries are recommended.

In [1], queries are rewritten based on a historical click graph in sponsored search. Given a query q, it first tries SimRank [18] to find similar queries to q. However, the authors found the cases where SimRank could fail in weighted click graph. For example, when two queries lead to clicks on two same ads rather than one ads, their similarity value would be even lower as measured by SimRank. Based on these observations, the authors develop two extended models based on SimRank to

measure query similarities for query rewriting. SimRank++ makes similarity scores to include evidence factor between nodes as well as weights of edges in the click graph and finds high proximity nodes using the historical click data. The evidence factor is defined as:

$$\text{evidence}(a, b) = \sum_{i=1}^{|E(a) \cap E(b)|} \frac{1}{2^i} \tag{6.8}$$

where $E(a)$ and $E(b)$ are the neighbors of node a and node b, respectively. The range of the evidence factor is $[0.5, 1]$. As the common neighbors increase, the evidence scores get closer to one. Let $s(a, b)$ denote the similarity metric from SimRank,

$$s(a, b) = \frac{C}{|E(a)| \cdot |E(b)|} \sum_{i \in E(a)} \sum_{j \in E(b)} s(i, j) \tag{6.9}$$

where the x and y are the two ads. The enhanced similarity scores including the evidence are designed as follows:

$$s_{evidence}(a, b) = evidence(a, b) \cdot s(a, b) \tag{6.10}$$

To support the weighted click graph, the authors make the extension to include the impact of weights as

$$s_{weighted}(a, b) = evidence(a, b) \cdot C \sum_{i \in E(a)} \sum_{j \in E(b)} W(a, i) W(b, j) s_{weighted}(i, j) \tag{6.11}$$

where $W(a, i)$ and $W(b, j)$ are functions of the weight set and its variance. The basic concept for SimRank and SimRank++ is that two objects are similar if they reference the same objects. The authors' work in [1] makes SimRank similarity scores more intuitive for the area of sponsored search and the two enhanced versions yield better query rewriting results.

In [15], a unified and discriminative model is proposed based on conditional random field (CRF) for query refinements on the morphological level. The proposed CRF-QR model involves different refinement tasks simultaneously, including spelling error correction, word merging, word splitting, and phrase segmentation. The authors designed two variants of the CRF-QR model, a basic model for single refinement task and an extended model for multiple refinement tasks. Let $\mathbf{x} = x_1 x_2 \ldots x_n$ and $\mathbf{y} = y_1 y_2 \ldots y_n$ denote a sequence of query words and sequence of refined query words, respectively. Let $\mathbf{o}$ denote a sequence of refinement operations.

Here n is the length of the sequence. The basic CRF-QR model is obtained as follows:

$$Pr(\mathbf{y}, \mathbf{o}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{n} \phi(y_{i-1}, yi)\phi(y_i, o_i, \mathbf{x}) \tag{6.12}$$

where $\phi(y_{i-1}, y_i)$ is the potential function showing the adjacent y's mutual dependence, and $\phi(y_i, o_i, \mathbf{x})$ is the potential function showing the dependence of y_i on the operation o_i and the input query $\mathbf{x}$. Z is the normalizing factor. The individual o's are independent from each other to simplify the model, because the dependency existing between o's has been captured by the dependency between y's. The space of the refined query y is as extremely large as the space of the original query x before introducing o into the model. An operation o can be insertion, deletion, and substitution of letters in a word or transposition. Because the space of operation o is very limited, the mapping from x's to y's under operation o is not completely free. o works as a constraint in the CRF-QR model to reduce the space of y for given x. When multiple refinement tasks are needed, the extended CRF-QR model uses multiple sequences of operations $\overrightarrow{o_i} = o_{i,1}o_{i,2}, \ldots, o_{i,m_i}$ and the corresponding sequences of intermediate results in $\overrightarrow{z_i} = z_{i,1}z_{i,2}, \ldots, z_{i,m_i}$.

$$Pr(\mathbf{y}, \overrightarrow{\mathbf{o}}, \overrightarrow{\mathbf{z}}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{n} (\phi(y_{i-1}, y_i) \prod_{j_i=1}^{m_i} \phi(z_{i,j_i}, o_{i,j_i}, z_{i,j_i-1})) \tag{6.13}$$

The prediction of the most likely refined query $\mathbf{y}^*$ satisfies

$$\mathbf{y}^*\mathbf{o}^* = \arg\max_{\mathbf{y},\mathbf{o}} Pr(\mathbf{y}, \mathbf{o}|\mathbf{x}) \tag{6.14}$$

The extended CRF-QF model can perform different query refinement tasks or operations simultaneously. Because the tasks are interdependent sometimes, the accuracy of tasks can be enhanced.

In [4], to solve the query rewriting problem, the authors leverage the query log data and follow the common procedure to generate some candidate queries first before using a scoring method to rank the quality of the candidate queries. Query term substitution is applied as the major approach for candidate query generation. Social tagging data is utilized as a helpful resource for extracting candidate substitution term. A graphical model taking into account the semantic consistency is designed for query scoring. The authors exploit the latent topic space of a graph model to assess the candidate query quality.

In addition to query reformulation and click graph data, anchor texts are used for the query rewriting problem in [8] that are often associated with links to documents. Since they are selected manually to describe the associated web documents, they provide very relevant information to these documents. It is demonstrated that anchor

texts usually offer more accurate description of their associated documents than the web documents themselves [6].

6.2.2 Translation-Based Methods

If we consider user queries as the source language and web documents as the target language, one natural way for query rewriting is to translate a source language of user queries into a target language of web documents [26, 27].

In [27], statistical machine translation (SMT) models have been adopted for query rewriting. The alignment template approach in [25] is adopted as SMT for query rewriting. It contains a translation model and a language model. It aims to seek the English string $\hat{e}$ as a translation of a foreign string f:

$$\hat{e} = arg \max_{e} p(e|f) = arg \max_{e} p(f|e)p(e) \tag{6.15}$$

where $p(f|e)$ is the translation model and $P(e)$ is the language model. $\hat{e}$ is further formulated as a combination of a set of feature functions $h_m(e, f)$ with the corresponding weight λ_m as:

$$\hat{e} = arg \max_{e} \sum_{m=1}^{M} \lambda_m h_m(e, f) \tag{6.16}$$

The translation model used in query rewriting is according to the sequence of alignment models [24]. A hidden variable is introduced to capture the relation between translation and alignment models for source string $f = f_1^J$ and target string $e = e_1^I$. The hidden variable is used to denote an alignment mapping from source position j to target position a_j:

$$P(f_1^J|e_1^I) = \sum_{a_1^J} P(f_1^J, a_1^J|e_1^I) \tag{6.17}$$

To align source words to the empty word, a_1^J includes null-word alignments $a_j = 0$. In the query rewriting, we adopt an n-gram language model which gives a string w_1^L of words with the following probability:

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{L} P(w_i|w_{i-n+1}^{i-1}) \tag{6.18}$$

A corpus of user queries is utilized to estimate the n-gram probabilities. A variety of smoothing techniques are used to mitigate the data sparse problems [5].

Once the SMT system is trained, to translate unseen queries, a standard dynamic-programming beam-search decoder is used [25].

In detail, pairs of user queries and snippets of clicked results are used as the parallel corpus and then a machine translation model is trained on the corpus. Once the model is trained, query rewriting is similar to the decoding process in machine translation. During decoding, a large n-gram language model is trained on queries. It is shown that the proposed method achieves improved query rewriting performance compared to methods based on term correlations.

The SMT system in [27] is used as a black box and it is hard to verify the contributions of its components. Thus, lexicon models have been utilized in [11]. There are two phases for the proposed framework: candidate generation and candidate ranking. In the phase of candidate generation, the original query is tokenized as a term sequence. For each non-stop word, a lexicon model is used to generate its translated words according to the word translation probabilities. In the candidate ranking phase, a ranking algorithm based on Markov random field (MRF) is used to rank all candidates. In this work, three lexicon models have been studied. The first lexicon model is the word model from IBM model 1 [7] which learns the translation probability between single words. The word model does not incorporate contextual information. Thus, the second lexicon model is a triplet model that uses triplets to incorporate word dependencies [16]. The third model is a bilingual topic model (BLTM) [10]. The intuition behind BLTM is through a query and its relevant documents can use different language styles or vocabularies, they should share similar topic distributions. The lexicon models are trained on pairs of queries and titles of clicked web documents. It is shown that the word model can generate rich candidates, and the triplet and topic models can select good expansion words effectively.

In [10], this paper provides a quantitative analysis of the language discrepancy issue and explores the use of clickthrough data to bridge documents and queries. We assume that a query is parallel to the titles of documents clicked on for that query. Two translation models are trained and integrated into retrieval models: A word-based translation model that learns the translation probability between single words and a phrase-based translation model that learns the translation probability between multi-term phrases. Experiments are carried out on a real-world dataset. The results show that the retrieval systems that use the translation models outperform significantly the systems that do not.

In [9], the authors follow [27] to consider the query rewriting problem as a machine translation problem and use pairs of queries and titles of clicked documents to train a machine translation model with the word model. Similar to [27], it shows that SMT based system outperforms systems based on term correlation. However, the word model considers isolated words while ignoring completely the context. It is observed that (1) consecutive words often form a phrase and (2) neighboring words can offer helpful contextual information. Thus, they introduce the constrained groups of term as concepts and propose concept-based translation models for query rewriting.

6.3 QRW with Deep Models

Deep learning techniques have powered a number of applications from various domains such as computer vision, speech recognition, and natural language processing. Recently, deep learning techniques have been adopted in the query rewriting task and in this section, we will review representative deep learning based query rewriting algorithms.

6.3.1 Word Embedding for QRW

In [14], the authors propose three models for query rewriting of sponsored search based on context and content-aware word embedding. The first model, context2vec, considers a query as a single word in a sentence and each query session as a sentence. Similar queries of similar context are supposed to have similar embeddings. The skip-gram model is used in this model for query representation learning by maximizing the objective function,

$$\mathcal{L} = \sum_{s \in S} \sum_{q_m \in s} \sum_{-b \leq i \leq b, i \neq 0} \log \mathbb{P}(q_{m+i} | q_m) \tag{6.19}$$

where S is the set of all search sessions, b is the window size of neighboring queries for the context, and $\mathbb{P}$ is the probability of observing a neighboring query q_{m+i} given the query q_m. The second model, content2vec, considers a query as a paragraph for word prediction without the session information. The word vectors are used to train the model for context words prediction within the query only by maximizing the objective function

$$\begin{aligned} \mathcal{L} = \sum_{s \in S} (& \sum_{q_m \in s} \log \mathbb{P}(q_m | \omega_{m1} : \omega_{mT_m}) \\ & + \sum_{w_{mt} \in q_m} \log \mathbb{P}(\omega_{mt} | \omega_{m,t-c} : \omega_{m,t+c}, q_m)) \end{aligned} \tag{6.20}$$

where c is the length of the context for words in the query. The vector representation for query q_m is $q_m = (\omega_{m1}, \omega_{m2}, \ldots \omega_{mT_m})$. A query's context should include both the content of the query and queries of the same session. The third model, context-content2vec, is a two-layer model combining the first two models and considering

both the search session context and the query context.

$$\mathcal{L} = \sum_{s \in S} \sum_{q_m \in s} \Big(\sum_{-b \le i \le b, i \ne 0} \log \mathbb{P}(q_{m+i}|q_m) + \alpha_m \log \mathbb{P}(q_m|\omega_{m1} : \omega_{mT_m}) + \sum_{\omega_{mt} \in q_m} \log \mathbb{P}(\omega_{mt}|\omega_{m,t-c} : \omega_{m,t+c}, q_m) \Big) \tag{6.21}$$

The models are trained on Yahoo search data including 12 billion search sessions.

6.3.2 Seq2Seq for QRW

As a neural sequence model, recurrent neural network (RNN) obtains the best performance on numerous important sequential learning tasks. The long short-term memory (LSTM), one of the most popular RNN variants, can capture long range temporal dependencies and mitigate the vanishing gradient problem. In the work [17], the authors adopt the sequence-to-sequence LSTM model to build a two-stage query rewriting frameworks [29].

In the first stage, the model training stage, the input sequence $x_1^J = x_1, \cdots, x_J$ is the original query and the target output sequence $y_1^J = y_1, \cdots, y_J$ is the rewritten queries. For the LSTM variant in the Seq2Seq model, the gates and cells are implemented by the following composite functions [13]:

$$\begin{cases} i_j = \sigma(W_{xi}x_j + W_{hi}h_{j-1} + b_i) \\ f_j = \sigma(W_{xf}x_j + W_{hf}h_{j-1} + W_{cf}c_{j-1} + b_f \\ c_j = f_j c_{j-1} + i_j tanh(W_{xc}x_j + W_{hc}h_{j-1} + b_c) \\ o_j = \sigma(W_{xo}x_j + W_{ho}h_{j-1} + W_{co}c_j + b_o) \\ h_j = o_j tanh(c_j) \end{cases} \tag{6.22}$$

where $h_1^J = h_1, \cdots, h_J$ is the hidden vector sequence, $W_{\cdot,\cdot}$ terms are the weight matrices, and $b_\cdot$ terms are the bias vectors.

The sequence-to-sequence LSTM aims to estimate the conditional probability $p(y_1, \cdots, y_I|x_1, \cdots, x_J)$, where $x_1, \cdots, x_J$ is an input sequence and $y_1, \cdots, y_I$ is its corresponding output sequence. The LSTM computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence $x_1, \cdots, x_J$ given by the last hidden state of the LSTM and then computing the probability of $y_1, \cdots, y_I$ with a standard LSTM–LM formulation whose initial hidden state is set as the representation v of $x_1, \cdots, x_J$:

$$p(y_1, \cdots, y_I|x_1, \cdots, x_J) = \prod_{i=1}^{I} p(y_i|v, y_1, \cdots, y_{i-1}) \tag{6.23}$$

where $p(y_i|v, y_1, \cdots, y_{i-1})$ is represented with a softmax over all the words in the vocabulary. It learns a large deep LSTM on large-scale query and rewrites query pairs. It is trained by maximizing the log probability of a correct rewrite query $r = rt_1, rt_2, \cdots, rt_n, < EOQ >$ given the query $q = qt_1, qt_2, \cdots, qt_m, < EOQ >$, where "$< EOQ >$" is a special end-of-query symbol. Thus the training objective is

$$\frac{1}{|D|} \sum (q, r) \in D \log p(r|q) \tag{6.24}$$

where D is the training dataset and $p(r|q)$ is calculated according to Eq. (6.23). Once training is complete, original queries are fed to the model and rewrite candidates are produced by finding the most likely rewrites according to the LSTM.

In the second stage, the prediction stage, a beam-search method is used to output the most probable sequences. It searches for the most likely query rewrites using a simple left-to-right beam-search decoder instead of an exact decoder. It maintains a small number B of partial rewrites, where partial rewrites are prefixes of some query rewrite candidates. At each time-step, it extends each partial rewrite in the beam with every possible word in the vocabulary. It discards all but the B most likely rewrites according to the model's log probability.

In [28], a novel method is proposed to translate a natural language query into a keyword query relevant to the natural language query, which can be applicable to legacy web search engines for retrieving better search results. Since legacy search engines are optimized for short keyword queries, a natural language query submitted directly to legacy search engines will highly likely lead to search results of low relevance. The proposed method introduces a RNN encoder–decoder architecture. To translate the input natural language query $\mathbf{x} = \{x_1, \cdots, x_n\}$ into a keyword query $\mathbf{y} = \{y_1, \cdots, y_m\}$, the RNN encoder–decoder models the conditional probability $p(\mathbf{y}|\mathbf{x})$ to complete the translation process. The encoder reads $\mathbf{x}$ sequentially to generate the hidden state. The decoder generates one keyword at a time by decomposing the probability of the keyword query $\mathbf{y}$ into conditional probabilities,

$$p(\mathbf{y}) = \prod_{i=1}^{m} p(y_i|y_1, \cdots, y_{i-1}, \mathbf{x}) \tag{6.25}$$

The prediction of the current keyword is based on the input $\mathbf{x}$ and the previously generated keywords. An attention mechanism is adopted to avoid biased representation caused by weakly relevant words in long natural language queries. The attention-based RNN encoder–decoder model is trained by maximizing the conditional log-likelihood as

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{m} \log p(y_i = y_i^{(j)}|y_1^{(j)}, \cdots, y_{i-1}^{(j)}, \mathbf{x}^{(j)}) \tag{6.26}$$

where $y_i^{(j)}$ is the i-th keyword of the j-th training instance in the training set. Since the training only needs pairs of a natural language query and its keyword query, the training is independent on the choice of the search engine and the proposed model can adapt legacy web search engines to natural language queries.

6.3.3 Learning to Rewrite Methods

In [17], a learning to rewrite framework is proposed that contains candidate generation and candidate ranking. The query rewriting problem aims to find the query rewrites of a given query for the purpose of improving the relevance of the information retrieval system. The proposed framework formulates the query rewriting problem as an optimization problem of finding a scoring function $F(q, r)$ which assigns a score for any pair of query q and its rewrite candidate r. The framework assumes that $\mathcal{G} = \{g_1, g_2, \ldots, g_M\}$ is a set of M candidate generators. Candidate generators could be any existing query rewriting techniques. In the candidate generating phase, we use candidate generators in $\mathcal{G}$ to generate a set of rewrite candidates for a given query q as $\mathcal{R} = \{r_1, \cdots, r_n\}$, where n is the number of generated rewrite candidates. Each pair of query q and its rewrite candidate r_i, i.e., (q, r_i), is scored by the function $F(q, r_i)$. The rewrite candidates from $\mathcal{R}$ are then ranked based on the scores $\{F(q, r_1), F(q, r_2), \ldots, F(q, r_n)\}$ in the candidate ranking phase. A key step of the learning to rewrite problem is how to obtain the scoring function F.

Let $\mathcal{F} = \{f : (q, r) \mapsto f(q, r) \in \mathcal{R}\}$ be the functional space of the scoring functions for any pair of query and rewrite candidate and $\mathcal{Q} = \{q_1, \cdots, q_m\}$ be a set of m queries. We use $\mathcal{R}_i = \{r_{i,1}, \cdots, r_{i,n_i}\}$ to denote the set of rewrite candidates of query q_i generated by $\mathcal{G}$, where n_i is the number of rewrite candidates for q_i. For each query q_i, we further assume that $\mathcal{I}_i$ is the learning target that encodes the observed information about the quality of rewrite candidates in $\mathcal{R}_i$. Note that the forms of I_i are problem-dependent that could be the label for each pair (q, r_i) or the preferences among $\mathcal{R}_i$ for q_i. With the aforementioned notations and definitions, the problem of searching in $\mathcal{F}$ for a scoring function $F(q, r)$ is formally stated as the following minimization problem:

$$F = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{m} L(f, q_i, \mathcal{R}_i, \mathcal{I}_i) \tag{6.27}$$

The exact forms of the loss function $L(f, q_i, \mathcal{R}_i, \mathcal{I}_i)$ depend on the learning target $\mathcal{I}_i$ and three types of loss functions are introduced including point-wise, pair-wise, and list-wise loss. Generating the learning target $\mathcal{I}_i$ is challenging especially for a large set of queries and their corresponding rewrite candidates. One straightforward way is to use human labeling. However, it is not practical, if not impossible, to achieve this for a web-scale query rewriting the application. First, it is very time and

effort consuming to label millions of query rewriting pairs. Second, the relevance performance depends on many components of a search engine such as relevance ranking algorithms, thus it is extremely difficult for human editors to compare the quality of rewrite candidates. Third, for a commercial search engine, its components are typically evolved rapidly and in order to adapt to these changes, human labels are consistently and continuously needed. Therefore it is desirable for an automatic approach to generate the learning target. In this work, we specifically focus on boosting the relevance performance via query rewriting, thus the learning target should indicate the quality of the rewrite candidates from the perspective of search relevance. Intuitively a better rewrite candidate could attract more clicks to its retrieved documents. In other words, the number of clicks on the returned document from a rewrite candidate could be a good indicator about its quality in terms of relevance. These intuitions pave us a way to develop an automatic approach to generate learning target based on the query-document click graph that is extracted from search logs.

In [31], a co-training framework is proposed for query rewriting and semantic matching based on the learning to rewrite framework in [17]. It first builds a huge unlabeled dataset from search logs, on which the two tasks can be considered as two different views of the relevance problem. Then it iteratively co-trains them via labeled data generated from this unlabeled set to boost their performance simultaneously. A series of offline and online experiments have been conducted on a real-world e-commerce search engine, and the results demonstrate that the proposed method improves relevance significantly.

6.3.4 Deep Reinforcement Learning for QRW

In [22], the authors propose a query rewriting system by maximizing the number of relevant documents returned based on a neural network trained with reinforcement learning. The original query and each candidate term t_i from either the original query q_0 or from documents retrieved using q_0 are converted to a vector representation by using a CNN or a RNN. Then the probability of selecting each candidate term is computed. The search engine is treated as a black box that an agent learns to use to retrieve terms to maximize the retrieval performance. Thus, an agent can be trained to use a search engine for a specific task. An extended model is introduced to sequentially generate reformulated queries to produce more concise queries based on RNN or LSTM. Rather than being queried for each newly added term, the search engine is queried with multiple new terms at each retrieval step for faster query reformulation.

In [23], methods are investigated to efficiently learn diverse strategies in reinforcement learning for query rewriting. In the proposed framework an agent consists of multiple specialized sub-agents and a meta-agent that learns to aggregate the answers from sub-agents to produce a final answer. Sub-agents are trained on disjoint partitions of the training data, while the meta-agent is trained on the

full training set. The proposed method makes learning faster, because it is highly parallelizable and has better generalization performance than strong baselines, such as an ensemble of agents trained on the full data.

6.4 Conclusion

Query rewriting is a key task in modern search engines and has attracted increasing attention in the last decade. The recent achievements of deep learning have further advanced this research topic. In this chapter, we roughly divided existing query rewriting algorithms according to the architectures they adopted to shallow and deep query writing. For shallow query rewriting, we discuss representative algorithms for substitution-based and translation-based methods. For deep query rewriting, we detail key algorithms for methods based on word embedding, Seq2Seq, learning to rewrite and deep reinforcement learning.

References

1. Ioannis Antonellis, Hector Garcia-Molina, and Chi-Chao Chang. Simrank++: query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment*, 1 (1): 408–421, 2008.
2. Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6 (2–3): 99–130, 1996.
3. Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern information retrieval*, volume 463. ACM Press New York, 1999.
4. Lidong Bing, Wai Lam, and Tak-Lam Wong. Using query log and social tagging to refine queries based on latent topics. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management*, pages 583–592, 2011.
5. Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867, 2007.
6. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30 (1–7): 107–117, 1998.
7. Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19 (2): 263–311, 1993.
8. Van Dang and W. Bruce Croft. Query reformulation using anchor text. In *Proceedings of the Third International Conference on Web Search and Data Mining*, pages 41–50, 2010.
9. Jianfeng Gao and Jian-Yun Nie. Towards concept-based translation models using search logs for query expansion. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1–10, 2012.
10. Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, pages 1139–1148, 2010.

11. Jianfeng Gao, Shasha Xie, Xiaodong He, and Alnur Ali. Learning lexicon models from search logs for query expansion. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 666–676, 2012.
12. Zhiguo Gong, Chan Wa Cheang, and Leong Hou U. Web query expansion by wordnet. In *International Conference on Database and Expert Systems Applications*, volume 3588, pages 166–175, 2005.
13. Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. 2012.
14. Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. Context- and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 383–392, 2015.
15. Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. A unified and discriminative model for query refinement. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 379–386, 2008.
16. Sasa Hasan, Juri Ganitkevitch, Hermann Ney, and Jesús Andrés-Ferrer. Triplet lexicon models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 372–381, 2008.
17. Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1443–1452, 2016.
18. Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, 2002.
19. Rosie Jones and Daniel C. Fain. Query word deletion prediction. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 435–436, 2003.
20. Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396, 2006.
21. Hang Li and Zhengdong Lu. Deep learning for information retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1203–1206, 2016.
22. Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 574–583, 2017.
23. Rodrigo Nogueira, Jannis Bulian, and Massimiliano Ciaramita. Multi-agent query reformulation: Challenges and the role of diversity. In *Deep Reinforcement Learning Meets Structured Prediction, ICLR 2019 Workshop*, 2019.
24. Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29 (1): 19–51, 2003.
25. Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational linguistics*, 30 (4): 417–449, 2004.
26. Stefan Riezler and Yi Liu. Query rewriting using monolingual statistical machine translation. *Computational Linguistics*, 36 (3): 569–582, 2010.
27. Stefan Riezler, Yi Liu, and Alexander Vasserman. Translating queries into snippets for improved query expansion. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 737–744, 2008.
28. Hyun-Je Song, A.-Yeong Kim, and Seong-Bae Park. Translation of natural language query into keyword query using a RNN encoder-decoder. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 965–968, 2017.
29. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

30. Xuanhui Wang and ChengXiang Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 479–488, 2008.
31. Rong Xiao, Jianhui Ji, Baoliang Cui, Haihong Tang, Wenwu Ou, Yanghua Xiao, Jiwei Tan, and Xuan Ju. Weakly supervised co-training of query rewriting and semantic matching for e-commerce. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 402–410, 2019.
32. Yang Xu, Gareth J. F. Jones, and Bin Wang. Query dependent pseudo-relevance feedback based on Wikipedia. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–66, 2009.
33. Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational intelligence magazine*, 13 (3): 55–75, 2018.