RIN: Reformulation Inference Network for Context-Aware Query Suggestion

Jyun-Yu Jiang
Department of Computer Science
University of California, Los Angeles
jyunyu@cs.ucla.edu

Wei Wang
Department of Computer Science
University of California, Los Angeles
weiwang@cs.ucla.edu

ABSTRACT

Search engine users always endeavor to reformulate queries during search sessions for articulating their information needs because it is not always easy to articulate the search intents. To further ameliorate the reformulation process, search engines may provide some query suggestions based on previous queries. In this paper, we propose Reformulation Inference Network (RIN) to learn how users reformulate queries, thereby benefiting context-aware query suggestion. Instead of categorizing reformulations into predefined patterns, we represent queries and reformulations in a homomorphic hidden space through heterogeneous network embedding. To capture the structure of the session context, a recurrent neural network (RNN) with the attention mechanism is employed to encode the search session by reading the homomorphic query and reformulation embeddings. It enables the model to explicitly captures the former reformulation for each query in the search session and directly learn user reformulation behaviors, from which query suggestion may benefit as shown in previous studies. To generate query suggestions, a binary classifier and an RNN-based decoder are introduced as the query discriminator and the query generator. Inspired by the intuition that model accurately predicting the next reformulation can also correctly infer the next intended query, a reformulation inferencer is then designed for inferring the next reformulation in the latent space of homomorphic embeddings. Therefore, both question suggestion and reformulation prediction can be simultaneously optimized by multi-task learning. Extensive experiments are conducted on publicly available AOL search engine logs. The experimental results demonstrate that RIN outperforms competitive baselines across various situations for both discriminative and generative tasks of context-aware query suggestion.

CCS CONCEPTS

• **Information systems** → **Query representation**; **Query suggestion**; **Query reformulation**;

KEYWORDS

Query reformulation; query suggestion; query embedding; recurrent neural network; query session modeling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM '18, October 22–26, 2018, Torino, Italy
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271808

ACM Reference Format:
Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation Inference Network for Context-Aware Query Suggestion. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271808

1 INTRODUCTION

Although search engines have already become indispensable in our daily life, the difficulty of deciding the ideal queries is everlasting. The search intents are often sophisticated while queries are usually not only short but also ambiguous [9]. As a consequence, in order to refine the search results, users have to articulate their information needs by reformulating the queries. The burden is on users. To make searching easier, most modern search engines turn to query suggestion that provides recommendations for next queries. Because users may be unfamiliar with the topics or the vocabulary for formulating queries, the concise suggestions can further accelerate search satisfaction. Moreover, query suggestion also benefits various applications such as query auto-completion.

To capture the search intents, it is intuitive to exploit the context information, including previous queries and click-through information in the same search session. The idea of utilizing the context has been widely studied in both query suggestion [5, 8, 20, 23, 35, 45] and query auto-completion [4, 10, 44]. Many conventional context-aware methods solely rely on query association or similarity between queries. For example, reused terms [4, 23, 44] and query co-occurrence [5, 20, 29, 35, 45] are advantageous to discover promising queries. However, query association and similarity based methods suffer from data sparsity. Even though some works [8, 9] attempt to alleviate the hardship by clustering queries based on click-through data, query clusters can still be too sparse to raise the roof of the limitations. In addition, queries in search sessions are issued successively, but the order of queries is generally ignored by query co-occurrence and similarity. The search sessions are not only diverse but also highly complicated, so methods considering only query association or similarity may fail in learning how users reformulate queries.

Fully understanding query reformulations is the grail of context-aware query suggestion because the context of search sessions are affected by reformulating queries to refine search results. To analyze reformulation behaviors, pioneer researchers categorized query reformulations into predefined strategies [13, 24]. More specifically, these reformulation strategies can be typecast into *syntactic* and *semantic* reformulations. Syntactic reformulations consist of predefined syntactic changes between queries such as adding terms,

deleting terms, and acronym expansion [6, 24]. The clear definitions of syntactic reformulation strategies come in handy to design features for machine learning models of query suggestions [27]. On the contrary, semantic reformulations incorporate strategies changing the meanings of queries such as generalization and specialization [2]. In practice, the semantic interoperability of queries is usually achieved by resorting to established ontologies, thereby exteriorizing semantic reformulations for query suggestion [2, 25, 26]. However, most of the reformulation-based methods are restricted and count on predefined reformulation strategies or assistance of ontologies, suffering from out-of-scope strategies and ambiguous queries.

The growth of deep learning, especially recurrent neural networks (RNNs) on sequences, provides the opportunity to generalize the use of reformulations in query suggestion. Followed by the success in machine translation [3, 47], sequence-to-sequence (seq2seq) models based on RNNs are adopted for query suggestion [14, 46]. The seq2seq models read the previously issued queries as a sequence, and then freely generate a sequence of terms as the suggested query. The terms of each query can be also treated as a sequence to derive information through another RNN [46]. Furthermore, since diverse search sessions increase the difficulty of learning reformulations, Dehghani et al. [14] decompose the generation of a query into reformulation strategies of appending and copying terms. However, there are a few shortcomings of existing deep learning based methods in query suggestion. First, although reformulations have been guided during generation [14], reformulations in the context still remain an open problem. The lack of understanding of previous reformulations can result in unsatisfactory query suggestions as shown in previous studies [2, 27]. Second, predefined strategies are too restricted to learn sophisticated reformulations that can be hard to decompose or even out of the scope. A better representation of reformulation is needed. Last but not least, although to some degree RNNs consider the transition between queries in the search session, models can still have a hard time learning reformulations because relationships between queries are sparse and implicit. As a consequence, existing RNN-based models [14, 46, 50] rely on additional features for discriminative query suggestion and hence often fail in predicting the intended query.

In this paper, Reformulation Inference Network (RIN) is proposed to address the limitations. More precisely, we focus on modeling reformulation behaviors for query suggestion with the reformulation representations capturing both syntactic and semantic relations. The system of homomorphic query embedding based on term embeddings is first introduced to preserve the syntactic property of reformulations. In addition, a heterogeneous network embedding model integrates click-through data in search logs into the foundational term embeddings to capture semantic reformulations. Simultaneously reading both the query and the recent reformulation for each step, RIN encodes the search session into a context vector by an RNN with the attention mechanism [3]. The reformulation inferencer then improves the capability of the context vector by predicting the next reformulation. Based on the context vector, multi-task learning is employed to train a query discriminator and a query generator with the reformulation inferencer. We then can utilize the model to suggest queries. Here we summarize our contributions in the following.

- To the best of our knowledge, this paper is the first work to model user behaviors based on queries and reformulations with homomorphic encoding that simultaneously preserves syntactic and semantic properties. The general and flexible representations can benefit the learning of how users reformulate queries along search sessions for query suggestion.
- We propose the framework RIN that deals with the data sparsity by inferring not only the intended query but also the next reformulation for query suggestion. More specifically, the reformulation representations enable the opportunity to leverage the knowledge across syntactically and semantically similar reformulations. When the model becomes more proficient in forecasting the next reformulations, the ability to discriminate and generate suggested queries can be also sharpened as well with multi-task learning.
- Experiments conducted on the publicly available AOL search engine logs demonstrate that RIN significantly outperforms existing methods for either discriminative or generative query suggestion. A study of parameter sensitivity then indicates the robustness of the proposed framework.

In the rest of this paper, we present the related work in Section 2 and formally define two different query suggestion tasks in Section 3. The framework RIN is then clearly described in Section 4. We finally show the experimental results in Section 5 and give some conclusions in Section 6.

2 RELATED WORK

2.1 Query Suggestion and Auto-Completion

To understand search intents behind queries, the context information including previous queries and click-through data is usually employed for query suggestion and query auto-completion. Most of the existing studies rely on query association and query similarity in the search session. For example, association rules [16] and co-occurrence [15, 23] can be mined and calculated for query suggestion. The connections between consecutive queries can be also learned by a query-flow graph [5] or Markov models [8, 20]. The cosine similarity [4, 23] and the edit distance [10] are popular metrics to recommend queries that are similar to the context. To deal with the problems of data sparsity, some works attempt to cluster queries into denser groups. For instance, a bipartite graph based on click-through data can be built for discovering queries with similar concepts [9, 34, 35, 51]. The word distributions of queries can be also utilized for EM clustering [18]. In addition to clustering, machine learning frameworks with statistical features can also partially alleviate the sparsity problem [40, 43, 44]. To learn how users reformulate queries, Jiang et al. [27] model syntactic reformulations based on predefined reformulation strategies. Well-established ontologies can also be leveraged to learn semantic reformulations [26]. Recently, Sordoni et al. [46] propose to suggest queries with a hierarchical RNN as the first study of query suggestion with deep learning. Dehghani et al. [14] then improve the approach by decomposing the generation process into two reformulation strategies. Wu et al. [50] take the implicit user feedback into account to better rank queries for suggestion.

However, most of the previous methods do not consider user formulations while the remaining methods rely on few predefined

reformulation strategies or established ontologies. Especially for the state-of-the-art RNN-based methods [14, 46, 50], all of them fail in solely predicting the intended queries and rely on other features. Our approach differs from these works in that reformulations are explicitly represented over the search session, thereby directly benefiting the model to predict the next reformulation without any additional feature.

2.2 Query Reformulation

Query reformulation is the process that users refine the preceding queries in order to obtain more satisfactory search results. Previous studies focus on determining and predicting reformulation strategies [13, 24]. These reformulation strategies can be analyzed in two aspects, including *syntactic* and *semantic* reformulations. The syntactic reformulations are the changes of terms between queries, such as adding and removing terms [6]. The semantic reformulations address the changes of topics behind queries, such as generalization and specialization of the concepts [2]. Most of the works attempt to manually define reformulation strategies based on the above two aspects. For example, Boldi et al. [5] design four predefined strategies of query transition while Huang and Efthimiadis [24] classify reformulations into 15 different types. However, predefined reformulation strategies can be too limited to describe reformulations in general and sparse data. In this work, we propose to illustrate reformulations with representations instead of relying on a set of predefined reformulation strategies.

In addition to query suggestion [27], understanding reformulations is also beneficial to many other applications. Lee et al. [33] determine the term effectiveness for improving the search quality. Based on reformulations in a search session, the search results can be further personalized [25]. Ren et al. [42] leverage the concept of query reformulation to understand conversation logs. All of these works demonstrate not only the effectiveness but also the robustness of learning reformulations.

2.3 Query and Reformulation Embedding

Although word embedding [36] has been well-studied and successfully employed in many NLP tasks, the journey of embedding techniques for IR has just begun. Clinchant and Perronnin [12] concatenate continuous word embeddings for representing queries and documents. Zheng and Callan [54] employ a weighted combination of word embeddings for better representations and more satisfactory search results. To directly optimize representations for relevance ranking, query embeddings can be learned by ranking models [52, 53]. However, none of these methods derives embeddings for demonstrating reformulations.

To learn the distributed representations of reformulations, Mitra [37] proposes the first and the only work to measure the reformulation embeddings by the vector between query embeddings based on a convolutional neural network. However, the approach has some shortcomings. First, other queries and click-through data in search sessions are not considered while deriving query embeddings. As a consequence, the semantic information in the context is unfortunately ignored. Second, the embedding system is not homomorphic. In other words, reformulations on queries will not directly reflect upon the corresponding embeddings. To address these problems, in this study, RIN models reformulations over search sessions by the homomorphic embeddings that preserve both syntactic and semantic information.

3 PROBLEM STATEMENT

In this section, we first formally define the objectives of this paper. A search session can be formally represented as a sequence of queries $\langle q_1, q_2, \cdots, q_L \rangle$ submitted successively by a single user within a time interval. Each query q is composed of a set of terms $T(q)$ and associated with the corresponding click-through information u, which is a set of clicked URLs. Suppose the user intends to submit a query q_{L+1} after the search context $\langle q_1, q_2, \cdots, q_L \rangle$. The two goals of this paper are listed as follows:

(1) **Discriminative Query Suggestion**: Given a set of candidate queries Q_{can}, we would like to give a ranking of candidates $q_{can} \in Q_{can}$ so that q_{L+1} ranks as high as possible. Some previous work [46] also call this task next-query prediction.
(2) **Generative Query Suggestion**: Different from the discriminative task, the generative task does not rely on candidate sets. With only the search context, generative query suggestion aims to generate a query q'_{L+1}, which is expected to be as similar as possible to the intended query q_{L+1} based on some similarity measures, such as cosine similarity and position independent word error rate [14].

For simplicity, the context is referred to as the search context $\langle q_1, q_2, \cdots, q_L \rangle$; the context length is the number of queires in the search context.

4 REFORMULATION INFERENCE NETWORK

In this section, we present the proposed framework, Reformulation Inference Network (RIN), for modeling queries in search sessions.

4.1 Framework Overview

Figure 1 demonstrates the general schema of RIN. The model mainly consists of four components, including query session encoder, reformulation inferencer, query discriminator, and query generator. Based on reformulation representations derived from homomorphic query embeddings, the query session encoder wraps the search session into a vector using a recurrent neural network and session-level attention. The reformulation inferencer plays the main role in RIN to infer the next query reformulation with the encoded vector from the query session encoder. Finally, to solve either the discriminative or generative task, the query discriminator and generator can be learned with the reformulation inferencer by multi-task learning.

4.2 Distributed Reformulation Representation

We first formally define how to represent reformulations in our model. For syntactic reformulations, some works designed several discrete features to measure reformulations [27]; other works simplified reformulations as a few basic syntactic operations, such as copying a term from the context and appending a new term [14]. For semantic reformulations, the topics of queries can be utilized to observe the changes of concepts behind queries [26]. In this paper, we propose to model reformulations from both perspectives.

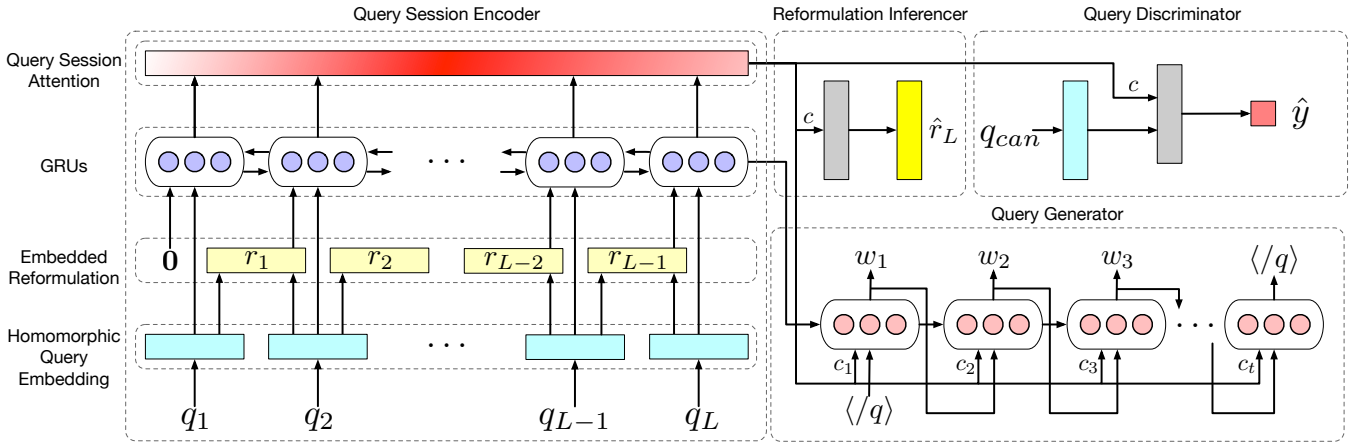

Figure 1: The schema of the proposed Reformulation Inference Network (RIN).

Inspired by homomorphic encryption that allows computations on ciphertexts [38], we want to design an embedding system that can reflect reformulations with subtraction computations. Theoretically, every reformulation can be syntactically factorized into adding and removing terms. Based on this concept, we come up with the *homomorphic query embedding* to represent each query as follows:

Definition 4.1 (Homomorphic Query Embedding). Suppose that every term t has a representative embedding $\boldsymbol{v}_t$. The homomorphic embedding of a query q is defined as:

$$\boldsymbol{v}_q = \sum_{t \in T(q)} \boldsymbol{v}_t,$$

where $T(q)$ consists of all terms in the query.

Based on the homomorphic query embeddings of queries, the reformulation $\boldsymbol{r_i}$ from q_i to q_{i+1} can be represented as the difference between embeddings as follows:

$$\boldsymbol{v}_{q_{i+1}} - \boldsymbol{v}_{q_i},$$

which can be also considered as a vector from q_i to q_{i+1} in the hidden space of homomorphic query embeddings. There are at least three benefits to apply homomorphic query embeddings to perform reformulations. First, the syntactic relations are homomorphically preserved as adding and subtracting term embeddings. Second, the latent space of embeddings also implicitly captures the semantic information of queries. Last but not least, the linear substructures of embeddings [36, 41] are helpful to understand the semantic relationships between reformulations and offer high interpretability. For example, the reformulation from "`Japan travel`" to "`Tokyo travel`" can be shown as $\boldsymbol{v}_{\texttt{Tokyo}} - \boldsymbol{v}_{\texttt{Japan}}$, which is close to $\boldsymbol{v}_{\texttt{Rome}} - \boldsymbol{v}_{\texttt{Italy}}$ from "`Italy hotel`" to "`Rome hotel`" under the country-capital reformulation substructure.

In this paper, we propose to exploit heterogeneous network embedding to learn the features of queries and terms. To learn the semantics of queries, term dependencies [32, 36, 41] and click-through data [26, 28, 34, 35] have been demonstrated to be useful. Here we propose to unify terms, queries, click-through data, and their relationships using a heterogeneous network. Figure 2 shows

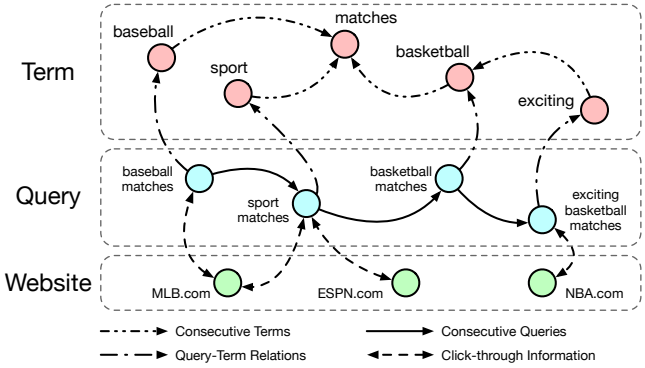

Figure 2: An example of the heterogeneous network constructed by a search session of four queries for deriving term embeddings. Note that the queries in the graph are auxiliary nodes connecting the domains of terms and websites.

an example of the network constructed by a search session of four queries. Each of terms, queries, and websites has a representative node in the network. Consecutive terms in each query and consecutive queries in each search session are connected. Each query also links to the first term and has bidirectional edges to the relevant websites clicked in the training logs. Finally, any of network embedding methods can be applied. Here node2vec [17], which is one of the state-of-the-art methods, is utilized to learn term embeddings as the base of homomorphic query embedding.

4.3 Query Session Encoder

The encoder of query sessions in RIN is a bidirectional recurrent neural network (Bi-RNN) with attention. The input of the encoder is a sequence $X = [x_1, x_2, \cdots x_L]$, where $x_i = [\boldsymbol{v}_{q_i}; \boldsymbol{r_{i-1}}]$ concatenates the homomorphic query embedding $\boldsymbol{v}_{q_i}$ and the reformulation $\boldsymbol{r_{i-1}}$ from the last query q_{i-1} for each query q_i in the search session. Note that $\boldsymbol{r_0}$ is set as a zero vector because the first query has no reformulation. The Bi-RNN reads the input sequence

twice as the forward pass and the backward pass. During the forward pass, the Bi-RNN creates a sequence of forward hidden states $\overrightarrow{\boldsymbol{h}} = [\overrightarrow{\boldsymbol{h_1}}, \overrightarrow{\boldsymbol{h_2}}, \cdots, \overrightarrow{\boldsymbol{h_L}}]$, where $\overrightarrow{\boldsymbol{h_i}} = \text{RNN}(\overrightarrow{\boldsymbol{h_{i-1}}}, x_i)$ is generated by a dynamic function such as LSTM [22] and GRU [11]. Here we use GRU instead of LSTM because it requires fewer parameters [30]. The backward pass processes the input sequence in reverse order. The backward hidden states are then generated as $\overleftarrow{\boldsymbol{h}} = [\overleftarrow{\boldsymbol{h_1}}, \overleftarrow{\boldsymbol{h_2}}, \cdots, \overleftarrow{\boldsymbol{h_L}}]$, where $\overleftarrow{\boldsymbol{h_i}} = \text{RNN}(\overleftarrow{\boldsymbol{h_{i+1}}}, x_i)$. Finally, the forward and backward hidden states are concatenated as the encoder hidden representations $\boldsymbol{h} = [\boldsymbol{h_1}, \boldsymbol{h_2}, \cdots, \boldsymbol{h_L}]$, where $\boldsymbol{h_i} = [\overrightarrow{\boldsymbol{h_i}}; \overleftarrow{\boldsymbol{h_i}}]$.

Since each query in the search session can have unequal importance for inferring the next query, the attention mechanism [3] is introduced to extract and aggregate representations that are more important than others. More specifically, the representation $\boldsymbol{h_i}$ will first be transformed by a fully-connected hidden layer to u_i to measure the importance α_i as follows:

$$\boldsymbol{u_i} = \tanh(\mathcal{F}_s(\boldsymbol{h_i})),$$
$$\alpha_i = \frac{\exp(\boldsymbol{u_i}^T \boldsymbol{u_s})}{\sum_{i'} \exp(\boldsymbol{u_{i'}}^T \boldsymbol{u_s})},$$

where $\mathcal{F}_s$ is the fully-connected hidden layer; tanh is chosen as the activation function for the convenience of similarity computation; $\boldsymbol{u_s}$ is a vector to measure the importance by computing $\boldsymbol{u_i}^T \boldsymbol{u_s}$. Finally, the normalized importance α_i is obtained through a softmax function. The context vector $\boldsymbol{c}$ can be represented as the weighted sum of the encoder hidden representations $\boldsymbol{h}$ as follows:

$$\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{h_i}.$$

4.4 Reformulation Inferencer

To enhance the capability of inferring the intended query, we assume that a model that accurately predicts the next reformulation can also correctly forecast the next query. More precisely, the intended query q_{L+1} is equivalent to the reformulation $\boldsymbol{r_L}$ with the last query q_L. Additionally, inferring reformulations is more trainable than directly predicting next queries because similar reformulations may be shared across different queries.

The aim of the reformulation inferencer is to predict the next reformulation $\boldsymbol{r_L} = \boldsymbol{v}_{q_{L+1}} - \boldsymbol{v}_{q_L}$ established by homomorphic query embeddings. Taking the context vector $\boldsymbol{c}$, the reformulation inferencer first applies a fully-connected hidden layer for the nonlinearity of prediction as follows:

$$\boldsymbol{u_r} = \text{ReLU}(\mathcal{F}_{h_r}(\boldsymbol{c})),$$

where ReLU$(\cdot)$ is the rectified linear unit [39] as the activation function; $\mathcal{F}_{h_r}(\cdot)$ is the fully-connected layer. The predicted reformulation $\hat{\boldsymbol{r_L}}$ can then be modeled as a linear combination of the result as follows:

$$\hat{\boldsymbol{r_L}} = \boldsymbol{W_r} \boldsymbol{u_r} + \boldsymbol{b_r},$$

where $\boldsymbol{W_r}$ and $\boldsymbol{b_r}$ are the weights and the biases for the combination. If the predicted reformulation $\hat{\boldsymbol{r_L}}$ is close to the actual reformulation $\boldsymbol{r_L}$, the model should also have considerable potential to infer the next query q_{L+1}.

4.5 Query Discrimination and Generation

The reformulation inferencer predicts the next reformulation, but the tasks mentioned in Section 3 cannot directly be solved by reformulations represented as homomorphic embeddings. Hence, *query discriminator* and *query generator* are proposed to directly solve discriminative and generative tasks, respectively.

Query Discriminator. Given a candidate query q_{can} and the context vector $\boldsymbol{c}$ from the query session encoder, the goal of the query discriminator is to assess how likely q_{can} is the intended query. More precisely, we want to predict a probabilistic score $\hat{y}$ to approximate the probability of being the intended query

$$y = P(q_{can} = q_{L+1} \mid \langle q_1, q_2, \cdots, q_L \rangle)$$

for each candidate query q_{can}.

The input of the query discriminator concatenates the homomorphic query embedding of q_{can} and the context vector $\boldsymbol{c}$ as $\boldsymbol{x_d} = [\boldsymbol{v}_{q_{can}}; \boldsymbol{c}]$. The probabilistic score $\hat{y}$ can then be generated by a sigmoid unit with a fully-connected hidden layer as follows:

$$\boldsymbol{u_d} = \text{ReLU}(\mathcal{F}_{h_d}(\boldsymbol{x_d})),$$
$$\hat{y} = \sigma(\mathcal{F}_d(\boldsymbol{u_d})),$$

where $\mathcal{F}_{h_d}(\cdot)$ and $\mathcal{F}_d(\cdot)$ are two fully-connected hidden layers; ReLU$(\cdot)$ is the activation function for the hidden layer; $\sigma(\cdot)$ is the logistic sigmoid function [19].

Query Generator. Without any candidate query, the query generator aims to produce a sequence of terms as the generated query q'_{L+1} that estimates the intended query q_{L+1}. Inspired by seq2seq [3, 47] in machine translation, the query generator is designed as a decoder to generate a sequence of terms based on the output of the query session encoder.

The query generator as a decoder also relies on RNN. To generate the t-th term w_t, the hidden state of RNN can be computed based on the last predicted query w_{t-1} as follows:

$$\boldsymbol{s_t} = RNN(\boldsymbol{s_{t-1}}, [w_{t-1}; \boldsymbol{c_t}]),$$

where $\boldsymbol{c_t}$ is the context vector of the t-th term; similar to the query session encoder, the dynamic function $RNN(\cdot)$ is GRU in the experiments. Instead of always using the general context vector $\boldsymbol{c}$, $\boldsymbol{c_t}$ estimates a more appropriate context vector because each query in the context may play different role in generating w_t. More precisely, the last hidden state of the decoder $\boldsymbol{s_{t-1}}$ is taken into account to compute the importance and construct the dynamic context vector $\boldsymbol{c_t}$ as follows:

$$\boldsymbol{u_{t,i}} = \tanh(\mathcal{F}_g([\boldsymbol{s_{t-1}}; \boldsymbol{h_i}])),$$
$$\alpha_{t,i} = \frac{\exp(\boldsymbol{u_{t,i}}^T \boldsymbol{u_g})}{\sum_{i'} \exp(\boldsymbol{u_{t,i'}}^T \boldsymbol{u_g})},$$
$$\boldsymbol{c_t} = \sum_i \alpha_{t,i} \boldsymbol{h_i},$$

where $\mathcal{F}_g(\cdot)$ is a fully-connected hidden layer. Based on $\boldsymbol{c_t}$, we further define the conditional probability for generating w_t as follows:

$$P(w_t \mid w_1, \cdots, w_{t-1}, \boldsymbol{c}) = f(\boldsymbol{s_t}),$$

where $f(\cdot)$ is a projection layer that estimates the conditional distribution over the vocabulary.

4.6 Learning and Optimization

The multi-task learning is applied to simultaneously train different components in RIN. Each of the reformulation inferencer, the query discriminator, and the query generator has a corresponding loss function jointly optimized with other components.

For the reformulation inferencer, the loss function loss_R optimizes the distance between the actual reformulation $\boldsymbol{r_L}$ and the predicted reformulation $\hat{\boldsymbol{r_L}}$ as follows:

$$\text{loss}_R = \frac{1}{2}||\boldsymbol{r_L} - \hat{\boldsymbol{r_L}}||_F^2,$$

where $||\cdot||_F$ is the Frobenius norm [49].

For the discriminative tasks, the query discriminator solves a binary classification problem. Hence the loss function loss_D focuses on reducing the binary cross-entropy [21] between the predicted probabilistic score $\hat{y}$ and the gold standard y as follows:

$$\text{loss}_D = -\left(y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right),$$

where y is a binary indicator demonstrating if the candidate query q_{can} is the intended query q_{L+1}.

To solve the generative tasks, the query generator produces a probability distribution over the vocabulary, so the loss function loss_G can be modeled by the cross-entropy between the generated sequences of words and the actual intended query as follows:

$$\text{loss}_G = -\sum_{w_t} \log P(w_t \mid S_t),$$

where w_t is the t-th term in the intended query, and S_t is the preceding terms of w_t.

Finally, the objective of multi-task learning combines the loss functions of different components as follows:

$$\text{loss} = \text{loss}_R + \text{loss}_{\text{task}},$$

where $\text{loss}_{\text{task}}$ can be either loss_D or loss_G based on the task to be solved. Moreover, the loss can also consider both of the loss functions, such as $\text{loss}_D + \text{loss}_G$, to simultaneously solve both problems.

5 EXPERIMENTS

In this section, we conduct extensive experiments and in-depth analysis to verify the performance and robustness of RIN for both discriminative and generative query suggestions.

5.1 Datasets and Experimental Settings

Datasets. We adopt the largest publicly available AOL search logs for experiments as previous studies [4, 26, 27, 44, 46, 50, 52, 53]. The 3-month dataset consists of queries submitted to the AOL search engine from 1 March, 2006 to 31 May, 2006. We first remove all non-alphanumeric characters and rare queries with less than ten occurrences in the whole log. The query logs are then segmented into sessions with a 30-minute threshold as the session boundary. Single-query sessions with only a single query are discarded because there must be at least one preceding query as the context for context-aware approaches. Every query with at least a preceding query as the context information is treated as the ground truth of the intended queries. The pre-processing process in this paper is consistent with previous studies [26, 27, 44]. To partition search sessions into training and testing sets, the first 2-month data are utilized for training while the remaining sessions are the testing data. Among the training data, 10% of sessions are randomly sampled as the validation set for parameter tuning. Finally, there are 1,357,500 training queries within 852,350 sessions and 647,559 testing queries within 403,772 sessions. Furthermore, to evaluate the performance using different context lengths, the testing set is partitioned into three subsets, including *Short Context* (1 query), *Medium Context* (2 to 3 queries), and *Long Context* (4 or more queries). As a result, Table 1 shows the statistics of queries with different context lengths in the training and testing datasets.

Table 1: The statistics of queries with different context lengths in the training and testing datasets.

Dataset	Context Length		
	Short (1 query)	Medium (2-3 queries)	Long (4+ queries)
Training	852,350	386,970	118,180
Testing	403,772	184,843	58,944

Evaluation. Similar to previous studies [14, 46, 50], the evaluation of discriminative query suggestion relies on candidate queries. The top-20 queries based on the frequency of the co-occurrences with the last query in the context are selected as candidate queries as *Most Popular Suggestion* (MPS) [14, 46] for re-ranking. The re-ranked results can then be evaluated by mean reciprocal rank (MRR). For generative query suggestion, we follow the previous study [14] to employ the position independent word error rate (PER) [48] to estimate the word overlap based similarity between the generated queries and the actual intended queries.

Implementation Details. The model is implemented by TensorFlow [1]. The Adam optimizer [31] is adopted to optimize the parameters and perform back-propagation algorithm based on gradients. The initial learning rate and the dropout parameter are set as 10^{-3} and 0.5. After the parameter tuning with the validation set, the number of hidden neurons for GRUs is set as 128, and the number of dimensions for homomorphic embeddings is 256.

Baseline Methods. To evaluate the performance of RIN, we compare with the following baseline methods in different categories.

- *Most Popular Suggestion* (MPS) [14, 46] is a maximum likelihood method, which relies on "wisdom of the crowd" and ranks queries by the co-occurrence to the last query in the context. Note that the candidate queries in the experiments are generated by MPS.
- *Query-based Variable Markov Model* (QVMM) [20] learns the probability of query transitions over sessions with the variable memory Markov model implemented by a suffix tree.
- *Hybrid Suggestion* (Hybrid) [4] considers both the context information and the popularity by ranking candidate queries based on a linear combination between the popularity (i.e., MPS in this paper) and the similarity to recent queries.
- In addition to the popularity of queries, *Personalized Completion* (PC) [44] also incorporates the personal query logs as long-term history to provide a personalized ranking model based on LambdaMART [7], which is one of the state-of-the-art ranking models.

Table 2: The MRR performance of different methods in the testing sets with different context lengths for the task of discriminative query suggestion. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.

Dataset	MPS [14, 46]	Hybrid [4]	PC [44]	QVMM [20]	RC [27]	HRED [46]	ACG [14]	RIN
Overall Context	0.5471	0.5823	0.5150	0.5671	0.6202	0.6207	0.6559	**0.8254**
Short Context (1 query)	0.5680	0.5822	0.5343	0.5862	0.5960	0.6100	0.6471	**0.8361**
Medium Context (2 to 3 queries)	0.5167	0.5841	0.4865	0.5338	0.6689	0.6489	0.6542	**0.8190**
Long Context (4 or more queries)	0.4826	0.5768	0.4575	0.5026	0.6704	0.6122	0.6669	**0.7611**

- *Reformulation-based Completion* (RC) [27] is the only non-deep learning baseline method exploiting query reformulation. 43 reformulation-based features are proposed to capture user reformulation behaviors over search sessions with LambdaMART.
- *Hierarchical Recurrent Encoder-Decoder* (HRED) [46] and *Seq2Seq with Copiers* (ACG) [14] are deep learning based query suggestion methods. HRED constructs a hierarchical encoder-decoder structure to model the sequential and hierarchical dependencies across terms and queries. ACG extends the seq2seq structure to read the terms in the search session and then learn whether to copy the used term or to add a new term.

Note that two deep learning baseline methods HRED and ACG solve the discriminative tasks by requiring an external feature set to train a LambdaMART model. Although our approach is also a deep learning based method, the query discriminator of RIN can address the task without any support of external features.

5.2 Experimental Results

Discriminative Query Suggestion. We first evaluate the performance for discriminative query suggestion. Table 2 shows the MRR performance of different methods over various context lengths. Note that a high MRR score indicates the actual intended queries are ranked more favorably.

The hybrid suggestion method (Hybrid) slightly boosts the suggestion performance of the popularity-based baseline method (MPS) by considering the similarity between candidate queries and the local context. On the contrary, the performance of the personalized completion method (PC) drops after considering the historical logs of users as long-term historical data. It is because an enormous amount of users in the search logs have only little or even no historical data so that the sparsity causes a severe over-fitting phenomenon. QVMM based on a variable-memory Markov model is also marginally better than MPS but not as well as Hybrid. The reason can be explained by the complicated search sessions that are too sparse to be modeled by query dependencies as shown by the lower performance with longer context. The reformulation-based completion method (RC) is the best non-deep learning method in the experiments. The promising results of the reformulation features used in RC show again that reformulations are helpful for modeling search sessions as previous studies [14, 27]. Two deep learning methods, HRED and ACG, outperform all of the other baseline methods because RNNs carefully capture the sequential information of terms and queries in search sessions. ACG is further better than HRED since two reformulation strategies are considered in the model. As the proposed approach in this paper, RIN surpasses

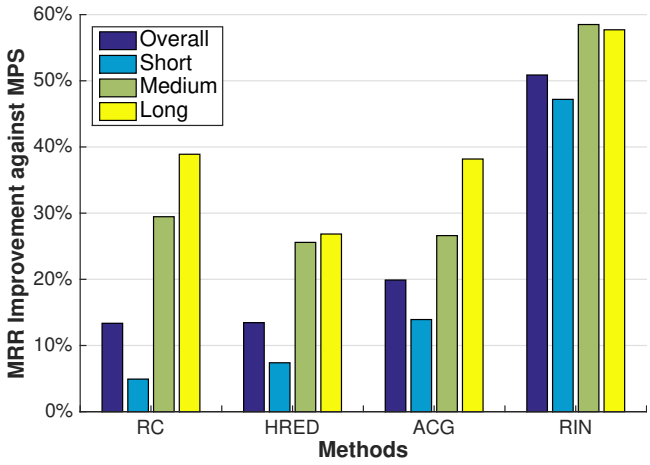

Figure 3: The MRR improvment of four methods over the MPS baseline method with different context lengths.

all of the baseline methods. More precisely, RIN achieves 50.87% and 25.83% improvements in the dataset of overall context over MPS and ACG, respectively. It is also worth noting that the improvements are consistent across all datasets with different context lengths. All of these improvements are significant at the 95% confidence level in a paired t-test.

To discuss the performance with different context lengths, we investigate the improvements of all methods over the naïve baseline method MPS because the performance on different datasets is not comparable. Figure 3 shows the improvements of three best baseline methods and RIN over MPS with different context lengths. It is reasonable to see the improvements are generally greater with longer contexts because sequential information and reformulations are more sufficient in longer sessions. An interesting observation is that the improvements of RIN over MPS are similar in the datasets of medium and long contexts. It demonstrates that RIN needs fewer queries to understand the search intents of users. The other observation is that RC outperforms HRED and achieves similar performance to that of ACG while there are multiple queries in the context. The results indicate reformulations are hard to be instinctively captured by RNNs. As a consequence, as shown in Table 2, HRED performs worse with long contexts, compared to other baselines considering reformulations as predefined features and strategies. Furthermore, RIN effectively learns reformulations and achieves the best performance.

Table 3: The PER performance of three methods. The improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.

Dataset	HRED [46]	ACG [14]	RIN
Overall	0.8069	0.6925	**0.6612**
Short (1 query)	0.8179	0.7015	**0.6851**
Medium (2 to 3 queries)	0.8338	0.6733	**0.6197**
Long (4 or more queries)	0.6753	0.6673	**0.6115**

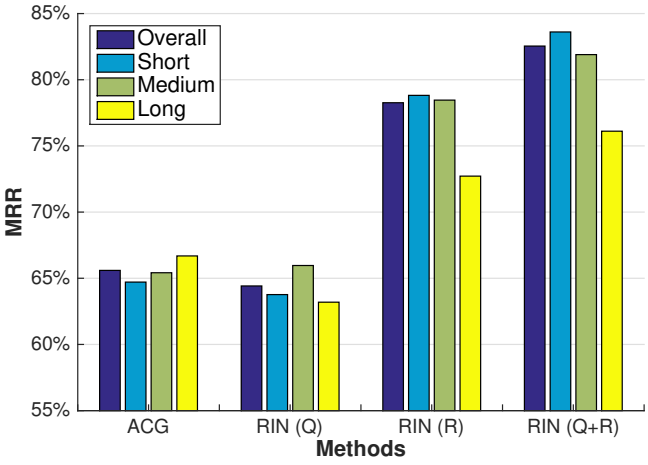

Figure 4: The MRR performance of RIN with either or both of homomorphic query and reformulation embeddings. Q denotes the homomorphic query embeddings while R indicates the embeddings of reformulations.

Generative Query Suggestion. Here we evaluate the performance for generative query suggestion. Note that only HRED and AGC are compared as the baseline methods because other methods are not generative models. PER as the evaluation metric treats each query as a bag of words and measures the difference between word sets. A low PER indicates high coverage of the predicted query to the intended query. Table 3 shows the PER performance of RIN and both baseline methods. Among two baseline methods, ACG outperforms HRED because of the consideration of reformulations. For HRED, it is interesting that PER dramatically improves when it comes to the long context dataset. It may be because longer contexts have more queries to capture the user behaviors. The proposed approach RIN outperforms both two baseline methods. More specifically, RIN obtains 22.02% and 4.72% improvements over HRED and ACG. For different context lengths, the improvements are greater with longer contexts. For instance, the improvement over ACG with short contexts is only 2.39%, but RIN improves ACG by 8.64% and 9.13% for the datasets of medium and long contexts, respectively. In addition, these results are also consistent with Figure 3 in the experiments of discriminative query suggestion.

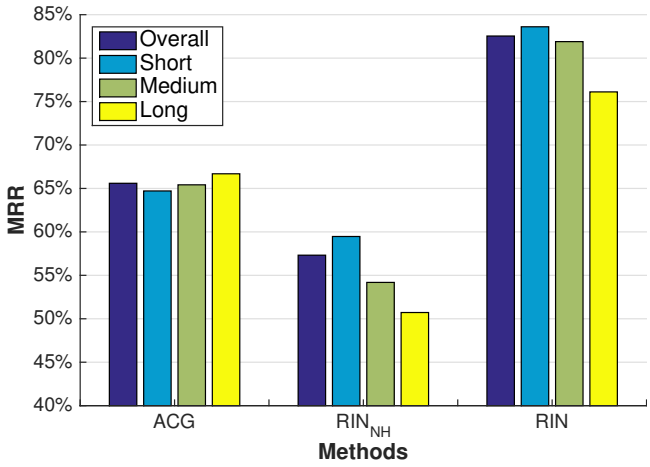

Figure 5: The MRR performance of RIN with homomorphic and non-homomorphic embeddings. Note that RIN_{NH} replaces the original homomorphic query embeddings with non-homomorphic query embeddings from node2vec.

5.3 Analysis and Discussions

In this section, we first analyze the effectiveness of the proposed model and then study the sensitivity of parameters.

Effectiveness of Homomorphic Embedding. We first investigate the effectiveness of the proposed homomorphic embedding of queries reformulations for query suggestion. As a leave-one-out analysis, we train RIN with only query embedding $\boldsymbol{v}_{q_i}$ or reformulation embeddings $\boldsymbol{r}_{i-1}$ to verify their individual capability. Figure 4 shows the MRR performance of RIN with and without homomorphic query and reformulation embeddings, where Q and R denote the use of query embeddings and reformulation embeddings. It is obvious that although the query embeddings are excluded, RIN (R) only lowers MRR by 5.18% from the comprehensive RIN (Q+R). However, when RIN (Q) only considers the query embeddings, the MRR drops by 21.96%. Moreover, RIN (Q) performs even worse than the best baseline method AGC. Even though reformulations are much beneficial for query suggestion, they are so hard to learn without directly being represented. The other observation is that the gain from the reformulation embeddings, i.e., the improvement of RIN (Q+R) over RIN (Q), is more substantial for shorter contexts. For example, MRR improves by 23.73% for the short context dataset while the improvement in the long context dataset is only 16.97%. When the reformulations are clearly given, it becomes more convenient for models to understand user behaviors and predict the intended queries.

Homomorphic vs. Non-homomorphic Embeddings. In addition to the effectiveness of homomorphic embeddings, we also want to show its advantage against non-homomorphic embeddings. Here we replace the query embeddings $\boldsymbol{v}_q$ in RIN with the query node embeddings from node2vec (see Section 4.2), which are non-homomorphic. Note that the reformulation embeddings will be calculated based on the updated query embeddings after the replacement, although they are not theoretically homomorphic. Figure 5

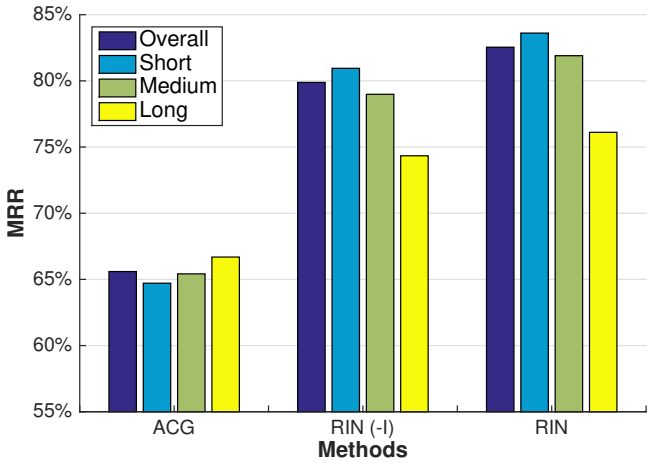

Figure 6: The MRR performance of RIN with and without the reformulation inferencer. Note that RIN (-I) removes the reformulation inferencer.

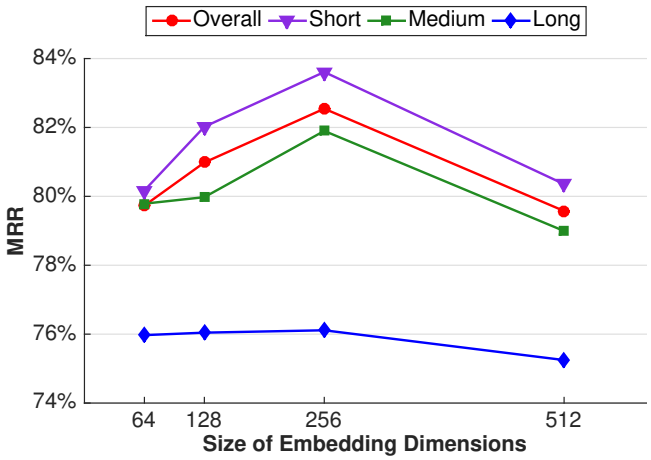

Figure 7: The MRR performance of RIN over different numbers of embedding dimensions.

shows the MRR performance of RIN with homomorphic and non-homomorphic embeddings. Note that RIN_{NH} represents RIN using non-homomorhpic embeddings. When the query embeddings of RIN become non-homomorphic, the MRR drops substantially. It is because non-homomorphic query embeddings fail in representing how users reformulate queries, especially for syntactic reformulations. Although the previous work [37] demonstrates some examples of syntactic relationships, non-homomorphic embeddings still cannot adequately preserve the syntactic reformulations.

Effectiveness of Reformulation Inferencer. Here we examine how the reformulation inferencer works in RIN. Similar to the previous analysis, we remove the component of reformulation inferencer and observe the performance. Figure 6 depicts the MRR performance of RIN with and without the reformulation inferencer, where RIN (-I) is trained without the inferencer component. After removing the component, the MRR performance decreases by 3.22%. It shows that the reformulation inferencer is actually helpful to predict the intended queries. Moreover, although the reformulation inferencer is ignored, RIN (-I) still outperforms the best baseline method ACG by 17.89% with reformulation embeddings as the inputs of query session encoder. Hence, the results also demonstrate the effectiveness of the proposed embedding method.

Size of Embedding Dimensions. Here we try to study how the size of reformulation embeddings (i.e., the nubmer of embedding dimensions) affects the performance. Figure 7 shows the MRR performance of RIN over different numbers of embedding dimensions. When the dimension number increases from a small size, MRR improves and peaks at the dimension number of 256. With a larger size of dimensions, RIN becomes overfitted because of the sparsity of search sessions. There is another interesting finding about long sessions. MRRs of datasets with shorter contexts are more sensitive to the embedding size than MRRs of the long context dataset. When the context consists of multiple queries, the embeddings of all queries and previous reformulations are fed into the model. Therefore, even though the embedding size is small, the combination of multiple queries in the RNN can still lead to a high degree of freedom for the prediction capability.

6 CONCLUSIONS

In this paper, we propose a novel approach for context-aware query suggestion by learning to represent query reformulations and model how users reformulate queries over search sessions. Inspired by homomorphic encryption, the *homomorphic query embedding* is introduced to reflect reformulations with computations based on semantic term embeddings. Our model, RIN, is then formulated as a neural network architecture to read the previous reformulations and infer the next reformulation in the embedding space, thereby addressing either discriminative or generative query suggestion. The extensive experiments demonstrate that our proposed model significantly outperforms seven baseline methods for both the discriminative and generative tasks of query suggestion. The improvements are consistent across different datasets of various context lengths. Moreover, the results of analysis also show the effectiveness and robustness of the proposed model. The reasons and insights can be concluded as follows: (1) reformulations are important to model search sessions, so homomorphic reformulations embeddings that precisely capture both syntactic and semantic reformulations can essentially benefit query suggestion; (2) the effectiveness of the reformulation inferencer in RIN implies that a model can be more capable of predicting the intended query if the next reformulation can be also anticipated; (3) longer contexts would not be necessarily required for understanding the search intents if the model can capture how users reformulate queries.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their helpful comments. The work was partially supported by NIH U01HG008488, NIH R01GM115833, NIH U54GM114833, and NSF IIS-1313606.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI '16*, Vol. 16. 265–283.

[2] Jun-ichi Akahani, Kaoru Hiramatsu, and Kiyoshi Kogure. 2002. Coordinating heterogeneous information services based on approximate ontology translation. In *Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems*. Citeseer, 10–14.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*. ACM, 107–116.

[5] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 56–63.

[6] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. 2009. From" dango" to" japanese cakes": Query reformulation models and patterns. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, Vol. 1. IEEE, 183–190.

[7] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[8] Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li. 2009. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th international conference on World wide web*. ACM, 191–200.

[9] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 875–883.

[10] Surajit Chaudhuri and Raghav Kaushik. 2009. Extending autocompletion to tolerate errors. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 707–718.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[12] Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. In *CVSC '13*. 100–109.

[13] Van Dang and Bruce W Croft. 2010. Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 41–50.

[14] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. In *CIKM '17*. ACM, 1747–1756.

[15] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *CIKM '05*. ACM, 696–703.

[16] Bruno M Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, and Nivio Ziviani. 2003. Using association rules to discover search engines related queries. In *Web Congress '03*. IEEE, 66–71.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD '16*. ACM, 855–864.

[18] Jiafeng Guo, Xueqi Cheng, Gu Xu, and Xiaofei Zhu. 2011. Intent-aware query similarity. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 259–268.

[19] Jun Han and Claudio Moraga. 1995. The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. In *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*. Springer-Verlag, 195–201.

[20] Qi He, Daxin Jiang, Zhen Liao, Steven CH Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web query recommendation via sequential query prediction. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 1443–1454.

[21] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[23] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the Association for Information Science and Technology* 54, 7 (2003), 638–649.

[24] Jeff Huang and Efthimis N Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 77–86.

[25] Di Jiang, Kenneth Wai-Ting Leung, and Wilfred Ng. 2011. Context-aware search personalization with concept preference. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 563–572.

[26] Jyun-Yu Jiang and Pu-Jen Cheng. 2016. Classifying User Search Intents for Query Auto-Completion. In *ICTIR '16*. ACM, 49–58.

[27] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *SIGIR '14*. ACM, 445–454.

[28] Jyun-Yu Jiang, Jing Liu, Chin-Yew Lin, and Pu-Jen Cheng. 2015. Improving ranking consistency for web search by leveraging a knowledge base and search logs. In *CIKM '15*. ACM, 1441–1450.

[29] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *WWW '06*. ACM, 387–396.

[30] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICML '15*. 2342–2350.

[31] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[32] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML '14*. 1188–1196.

[33] Chia-Jung Lee, Ruey-Cheng Chen, Shao-Hang Kao, and Pu-Jen Cheng. 2009. A term dependency-based approach for query terms ranking. In *CIKM '09*. ACM, 1267–1276.

[34] Zhen Liao, Daxin Jiang, Enhong Chen, Jian Pei, Huanhuan Cao, and Hang Li. 2011. Mining concept sequences from large-scale search logs for context-aware query suggestion. *TIST* 3, 1 (2011), 17.

[35] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. 2008. Query suggestion using hitting time. In *CIKM '08*. ACM, 469–478.

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS '13*. 3111–3119.

[37] Bhaskar Mitra. 2015. Exploring session context using distributed representations of queries and reformulations. In *SIGIR '15*. ACM, 3–12.

[38] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 113–124.

[39] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML '10*. 807–814.

[40] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: a machine learning framework for ranking query suggestions. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 25–34.

[41] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[42] Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. 2018. Conversational Query Understanding Using Sequence to Sequence Modeling. In *WWW '18*. International World Wide Web Conferences Steering Committee, 1715–1724.

[43] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2013. Learning to rank query suggestions for adhoc and diversity search. *Information Retrieval* 16, 4 (2013), 429–451.

[44] Milad Shokouhi. 2013. Learning to personalize query auto-completion. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 103–112.

[45] Yang Song, Dengyong Zhou, and Li-wei He. 2012. Query suggestion by constructing term-transition graphs. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 353–362.

[46] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM '15*. ACM, 553–562.

[47] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS '14*. 3104–3112.

[48] Christoph Tillmann, Stephan Vogel, Hermann Ney, Arkaitz Zubiaga, and Hassan Sawaf. 1997. Accelerated DP based search for statistical translation. In *EUROSPEECH '97*.

[49] Charles F Van Loan. 1996. Matrix computations (Johns Hopkins studies in mathematical sciences). (1996).

[50] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query Suggestion with Feedback Memory Network. In *WWW '18*. International World Wide Web Conferences Steering Committee, 1563–1571.

[51] Wei Wu, Hang Li, and Jun Xu. 2013. Learning query and document similarities from click-through bipartite graph with metadata. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 687–696.

[52] Hamed Zamani and W Bruce Croft. 2016. Estimating embedding vectors for queries. In *ICTIR '16*. ACM, 123–132.

[53] Hamed Zamani and W Bruce Croft. 2017. Relevance-based Word Embedding. In *SIGIR '17*. ACM, 505–514.

[54] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *SIGIR '15*. ACM, 575–584.