Click Feedback-Aware Query Recommendation Using Adversarial Examples

Ruirui Li
UCLA
rrli@cs.ucla.edu

Liangda Li
Yahoo Research
liangda@yahoo-inc.com

Xian Wu
University of Notre Dame
xwu9@nd.edu

Yunhong Zhou
Yahoo Research
yunhongz@verizonmedia.com

Wei Wang
UCLA
weiwang@cs.ucla.edu

ABSTRACT

Search engine users always endeavor to formulate proper search queries during online search. To help users accurately express their information need during search, search engines are equipped with query suggestions to refine users' follow-up search queries. The success of a query suggestion system counts on whether we can understand and model user search intent accurately. In this work, we propose Click Feedback-Aware Network (*CFAN*) to provide feedback-aware query suggestions. In addition to modeling sequential search queries issued by a user, *CFAN* also considers user clicks on previous suggested queries as the user feedback. These clicked suggestions, together with the issued search query sequence, jointly capture the underlying search intent of users. In addition, we explicitly focus on improving the robustness of the query suggestion system through adversarial training. Adversarial examples are introduced into the training of the query suggestion system, which not only improves the robustness of system to nuisance perturbations, but also enhances the generalization performance for original training data. Extensive experiments are conducted on a recent real search engine log. The experimental results demonstrate that the proposed method, *CFAN*, outperforms competitive baseline methods across various situations on the task of query suggestion.

CCS CONCEPTS

• **Information systems → Query suggestion**; **Learning to rank**.

KEYWORDS

Query recommendation; context-aware; adversarial example

ACM Reference Format:
Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. 2019. Click Feedback-Aware Query Recommendation Using Adversarial Examples. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313412

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313412

1 INTRODUCTION

The effectiveness of keyword-based search engines depends largely on the ability of a user to formulate expressive search queries. Expressive search queries clearly and unambiguously describe users' search intents and bring users directly to the desired information. In practice, translating human thoughts into concise sets of keywords to form queries is never straightforward [2]. This is particularly true for search engine users, who are mostly untrained casual users. In many cases, casual users have very limited background knowledge about the information they are searching for. To assist users in formulating queries, modern search engines are equipped with query suggestions [5, 16, 17, 20]. Given a user query, a query suggestion system deduces the search intent of the user by recommending a set of queries that are more expressive than the original user input. Our goal is to investigate the key concerns of existing query suggestion systems and to propose a novel approach to improve the suggestion performance.

Web search queries are usually very short, typically with only one or two keywords each [13, 24]. Short queries lead to the ambiguity issue. The ambiguity weakens the expressiveness of queries because the search engine may not understand the users' hidden search intents. This results in poor rankings of retrieval results and jeopardizes user experiences consequently.

A successful query suggestion system depends on understanding and modeling user search intents accurately. The user search intents lie in interactions, i.e., searches and clicks, between users and search engines. These interactions are generally partitioned into groups to form search sessions based on the time when they happened. Each search session is driven by consecutive related search queries and clicks of search engine users. In a search session, a user refines his/her original query and submits a sequence of follow-up queries to pinpoint his/her information need. Between two issued queries, the user may also click suggested queries to explore. The issued queries, together with the clicks, jointly allow us to accurately understand users' hidden search intents and come up with appropriate queries as suggestions. Unfortunately, most existing works merely consider clicked suggestions as feedback when modeling user search intents.

To conduct feedback-aware query suggestions, we mine search logs. A search log contains historical records, each of which registers the details of a web search conducted by a user. Table 1 shows some sample records extracted from the search log of a real search engine. Each record includes a query string, an anonymous user ID by whom the query was formulated, query submission time, the

Table 1: Sample search log records

UserID	Query String	Time	Clicked Suggestion	Suggested Queries
A	apple	2018-07-01 09:10:01	apple stock	apple store, **apple stock**, apple.com, apple iphone xs max
				apple id, itunes, icloud, google maps
A	apple price	2018-07-01 09:11:03	apple price per share	apple price targets, iphone apple price, **apple price per share**, apple
				apple fruit price, apple stock price, does apple price match, apple price today
B	disney	2018-07-03 12:12:02	disney cartoons	walt disney world, disney world tickets, disney world, disney experience
				disney orlando, **disney cartoons**, disney store, disney junior
B	disney shows	2018-07-03 12:22:20	disney channel cartoons	disney shows list, disney xd shows, disney tv shows, popular disney shows
				disney channel cartoons, list of disney tv series, 2018 disney channel shows

query suggestion subsequently clicked by the user (if done), and the suggested queries shown on the search page with the clicked ones highlighted in bold. The sample records show two search sessions. In the first session, user A first submitted query "apple" to the search engine, and then clicked the second query suggestion "apple stock" in the suggestion list. After that, the user issued the second query "apple price", and clicked the suggestion "apple price per share". The second search session involved a different user, B. S/he started searching with the query "disney", and then clicked "disney cartoons" in the suggestion list. He/she continued the search with the query "disney shows" and clicked "disney channel cartoons" subsequently. From the first search session, we can see that the clicked suggestion "apple stock", as feedback of the user, compensates for the ambiguity of the second issued query "apple price". It provides us with extra information to infer the underlying search intent of the user, which is to know the stock price of Apple instead of the price of Apple electronic products, such Apple iPhone. The clicked suggestion "disney cartoons" in the second search session also implicitly indicates that the user was interested in disney cartoons shows when the user search for "disney shows". These observations motivate us to include user feedback when modeling the search intents of search engine users.

Various research works have been carried out to tackle the task of query suggestion. Among them, neural network based models have made impressive progress over the past few years [7, 20, 25]. Deep neural network models mimic the learning process of human brains. Training these neural network models generally requires a large amount of training data to achieve excellent performance. However, for search related tasks, search queries involved are very sparse, and generally follow a long-tail distribution. Thus, those tail queries, with low probabilities in the data distribution, lack abundant training supports. In addition, the suggested queries are highly relevant to each other in most cases because they are all related to the search query in semantics. Distinguishing suggestions to be clicked from the ones that won't be clicked and ranking them take extreme efforts. Therefore, there tends to be a great performance gap between training and test. The trained model can be very sensitive to unseen perturbed queries. This is because queries are very short and a subtle perturbation on a query can lead to different or even the opposite search intent. In general, human perception and cognition are robust to a vast range of nuisance perturbations. However, neural networks are currently far from achieving the same level of tolerance of such perturbations [21]. This motivates us to investigate the robustness property of modern query suggestion systems and propose an appropriate method to achieve high tolerance.

Before we proceed to describe the details of the proposed approach, we enumerate the key properties that a good query suggestion system should observe:

- **[Context-Awareness]** The recommender should be aware of the sequential search queries issued by the user. These sequential queries implicitly inform the search intent of the user.
- **[Feedback-Awareness]** The recommender should also be aware of user clicks. User clicks, as an additional information source, can potentially reduce query ambiguities and allow us to understand user search intent comprehensively and more accurately.
- **[Robustness]** The recommender should increase the generalization of the original training data and improve the resistance to nuisance perturbations to the extent as extreme as possible. This reduces the performance gap between training and test, and provides favorable rankings of suggestions robustly.

2 RELATED WORK

Deep neural networks have been applied to many prediction and recommendation tasks successfully [11, 26, 27]. It also demonstrated excellent performances in query suggestion tasks [7, 15, 20, 25]. HRED [20], as the first study, employs a hierarchical encoder-decoder model to achieve context-aware query suggestion. [7] introduces the copy mechanism in addition to employing the encoder-decoder framework to generate suggested queries. [25] employs a feedback memory network to model titles of user clicked URLs. Reformulation inference network (RIN), proposed in [15], explicitly incorporates query reformulations during training.

Our work differs from the above neural network based works in that previously user clicked suggestions are explicitly incorporated as feedback. In addition, adversarial examples [6, 8] are dynamically constructed during the training of *CFAN*, which not only significantly improve the robustness of the model but also dramatically enhance the generic ranking performance.

3 PROBLEM STATEMENT

In this section, we formally define the objective of this paper. A search sequence S, represented as a sequence of search queries $<Q_1^S, Q_2^S, ..., Q_{M_1}^S>$, is submitted successively by a single user within a time interval. Each query in the search query is associated with corresponding click-through information (if happens), which is a set of clicked suggestions for that query. A clicked suggestion sequence C, $<Q_1^C, Q_2^C, ..., Q_{M_2}^C>$, is formed by ordering the clicked suggestions increasingly by the time the clicks happens. The goal of

this paper is: Given a set of candidate queries $\boldsymbol{Q}_{\text{can}}$, we would like to generate a ranking of candidates, with each candidate $Q_{\text{can}} \in \boldsymbol{Q}_{\text{can}}$, so that the ones to be clicked rank as high as possible.

4 METHODOLOGY

In this section, we discuss how to make query suggestions based on the search and click interactions between users and search engines. To achieve effective and efficient query suggestions, we decompose the task into two components. First, based on the search query, we generate a set of relevant queries as suggestion candidates. These suggestion candidates are informative to cover different interpretations and aspects of the search query. Second, we rank the suggestion candidates based user's entire search query sequence and clicked feedback. Only the top ones are selected and shown to the user as query suggestions.

4.1 Candidate Generation

Given a search query Q, the candidate generation component is responsible for constructing an informative set of expressive queries $\boldsymbol{Q}_{\text{can}}$ as suggestion candidates. An expressive suggestion candidate clearly and unambiguously describes the search intent of the user. An informative set of suggestion candidates are queries, that are diversified to cover different interpretations and aspects of the search query.

To generate such diversified informative and expressive suggestion candidates, multiple sources can be incorporated, such as search sessions and contextual information in search logs, related keywords from trending news, etc. In this work, we incorporate two sources based on the search log. The first source generates suggestion candidates based on query dependencies in search sessions. Given a search query, the follow-up search queries in the same search session tend to be more expressive and informative. Therefore they are adopted as suggestion candidates. Another source for candidate generation is based on prefix matching between the search query and historical search queries in the search log.

After generating suggestion candidates, the candidates are further ranked based on a combination of statistical features, i.e., the total number of submissions within last week, last month, and last year. Only top suggestion candidates are kept for further ranking in the candidate ranking component (see Section 4.2), while others are filtered out. The purpose of filtering out these less frequent suggestion candidates is to not only generate popular and fresh candidates, but also guarantee that recommendation services respond in real time. In a nutshell, given an input query Q, the candidate generation component returns a set of relevant queries as suggestion candidates.

4.2 Candidate Ranking

In this section, we discuss how we rank the suggestion candidates by mining the historical interactions between users and search engines. To better explain the ranking process, we first discuss the construction of training instances. Then, we go into details on how to utilize these instances to train *CFAN*.

Training instances are constructed from the historical search logs. Each instance contains a sequence of user issued queries S, a sequence of user clicked queries C, and a candidate suggested query Q_{can}. Two types of instances are constructed, i.e., positive instances and negative instances, denoted as Ins^+ and Ins^-, respectively. A positive instance is constructed if S and C jointly lead to the click of Q subsequently, while a negative instance is constructed if a query $\tilde{Q}$ is not clicked after S and C. In particular, to generate strong negative instances, only query $\tilde{Q}$, shown together with Q after searching the last query in S, is considered as a candidate query in the construction of negative instances. Positive instances Ins^+ and negative instances Ins^- jointly define the overall matching and ranking performance among a search query sequence S, a clicked suggestion sequence C, and a set of suggestion candidates $\boldsymbol{Q}_{\text{can}}$. We further define such a set of Ins^+ and Ins^- as a training view, denoted as V.

The clicks in the constructed training views allow us to tap into the wisdom of the crowds and provide a more favorable ranking of suggestions when running into the same or similar search and click scenarios. Given such scenarios, we expect the previously clicked suggestions rank ahead of those unclicked ones. To achieve this goal, Equation 1 shows the ranking loss function in *CFAN* over one training view V.

$$\text{Loss}_{\text{ranking}}(\Theta) = -\log\left(1 + \frac{\sum\limits_{\mathbf{ins}^+ \in V} g_\Theta(\mathbf{ins}^+)}{\sum\limits_{\mathbf{ins}^+ \in V} g_\Theta(\mathbf{ins}^+) + \sum\limits_{\mathbf{ins}^- \in V} g_\Theta(\mathbf{ins}^-)}\right), \tag{1}$$

where Θ is a set of parameters defining the ranking model, and $g_\Theta(\mathbf{ins})$ is a scoring function, which gives the ranking score of an instance. We use $\mathbf{ins}/\mathbf{ins}^+/\mathbf{ins}^-$ to represent the embedding of a general/positive/negative instance, respectively. By minimizing Equation 1 over all training views, $g_\Theta(\mathbf{ins}^+)$ is trained to have a high ranking score for clicked suggestions, while for unclicked suggestions, $g_\Theta(\mathbf{ins}^-)$ is trained to have a low score.

4.2.1 Search Intent Encoder. As we mentioned in the introduction, a user's search queries and clicked queries jointly help pinpoint his/her information need. A good search intent encoder is expected to incorporate both of them when modeling search intents. Note that both user search queries and clicked queries are inherently a sequence of queries; therefore, we first discuss how to model a single query through a query encoder. Then, we move on to the discussion of sequential query modeling through a query sequence encoder.

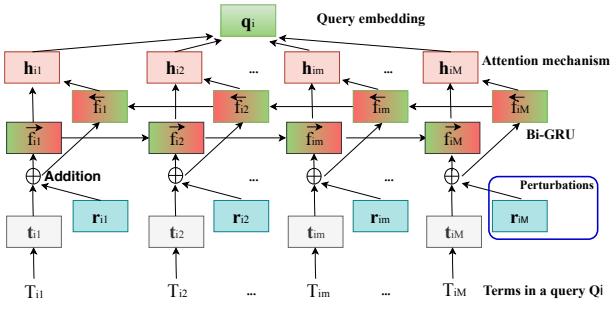

Figure 1: Query encoder with perturbed embeddings.

[Query Encoder.] Generally, queries contain different numbers of terms. In addition, different terms in a query contribute unequally

to the meaning of the query. For example, to search for the height of Barack Obama, users may formulate queries such as "how tall is Barack Obama", "height of Barack Obama", or even "Obama height". These three queries contain five, four, and two terms, respectively. Consider term importance in the first query, "tall", "Barack", and "Obama", these three terms, contribute more to understand the search intent of the query than the other terms, i.e., "how" and "is", which are generally less informative. In this work, we apply bidirectional GRU [18] and attention mechanism [22] to derive embeddings for a query Q.

Figure 1 shows the architecture of the query encoder. Given a query Q_i, composed of a sequence of terms $<T_{i1}, ..., T_{im}, ..., T_{iM}>$, where M is the number of terms in Q_i, we first embed each term T_{im} to an vector $\boldsymbol{t}_{im}$ through an embedding matrix as shown in Equation 2. To train robust models, adversarial examples are introduced by adding perturbed embeddings $\boldsymbol{r}$ into $\boldsymbol{t}$ (See more details in Section 4.2.3). We then use a bidirectional GRU to get the query embedding by summarizing information from both directions for terms in a query. The bidirectional GRU contains a forward GRU $\overrightarrow{f}$ which reads query Q_i from T_{i1} to T_{iM} and a backward GRU $\overleftarrow{f}$ which reads from T_{iM} to T_{i1}:

$$\boldsymbol{t}_{im} = Emb_t(T_{im}), m \in [1, M]. \tag{2}$$

$$\overrightarrow{\boldsymbol{f}}_{im} = \overrightarrow{GRU}(\boldsymbol{t}_{im}), m \in [1, M]. \tag{3}$$

$$\overleftarrow{\boldsymbol{f}}_{im} = \overleftarrow{GRU}(\boldsymbol{t}_{im}), m \in [M, 1]. \tag{4}$$

We obtain a representation $\boldsymbol{h}_{im}$ for each term T_{im} in query Q_i by concatenating the forward hidden state $\overrightarrow{\boldsymbol{h}}_{im}$ and backward hidden state $\overleftarrow{\boldsymbol{h}}_{im}$, i.e., $\boldsymbol{h}_{im} = [\overrightarrow{\boldsymbol{h}}_{im}; \overleftarrow{\boldsymbol{h}}_{im}]$, which summarizes the information of the whole query but centered around term T_{im}.

Since not all terms in a query contribute equally to the search intent embedded in the search query, attention mechanism is applied to extract such informative contributing terms and aggregate the embeddings of these terms to construct a query vector.

$$u_{im} = tanh(\boldsymbol{W}_w \boldsymbol{h}_{im} + b_w). \tag{5}$$

$$\alpha_{im} = \frac{\exp(u_{im}^T u_w)}{\sum_m \exp(u_{im}^T u_w)}. \tag{6}$$

$$\boldsymbol{q}_i = \sum_m \alpha_{im} \boldsymbol{h}_{im}. \tag{7}$$

Equations 5, 6, and 7 show the attention process on the query term level. The term embedding $\boldsymbol{h}_{im}$ is first fed into a one-layer MLP to get u_{im}. Then, u_{im} is further used to derive the normalized importance weight α_{im} through a softmax function. Finally, the query vector $\boldsymbol{q}_i$ is computed as a weighted sum of the term embeddings based on the weights.

[Query Sequence Encoder.] A query sequence is composed of a sequence of queries $<Q1, ...,Q_i, ..., Q_N>$, where N is the number of queries in the sequence. Both user search queries and clicked queries are essentially query sequences. Given a query sequence $<Q1, ...,Q_i, ..., Q_N>$, the query sequence encoder is expected to encode the hidden search intent in it. Similar to the structure of query encoder, the query sequence encoder contains a bidirectional GRU layer and an attention layer. The differences are two-fold. First, the query sequence encoder takes query embeddings as inputs. Second, there are no perturbations in query sequence encoder. Formally, a bidirectional GRU is first applied to encode the query sequence:

$$\overrightarrow{\boldsymbol{f}}_i = \overrightarrow{GRU}(\boldsymbol{q}_i), i \in [1, N], \tag{8}$$

$$\overrightarrow{\boldsymbol{f}}_i = \overleftarrow{GRU}(\boldsymbol{q}_i), i \in [N, 1], \tag{9}$$

The representation of a query Q_i in the sequence is constructed by concatenating $\overrightarrow{\boldsymbol{f}}_i$ and $\overrightarrow{\boldsymbol{f}}_i$, i.e., $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$. $\boldsymbol{h}_i$ summarizes the neighbor queries around query Q_i but still focus on the meaning of query Q_i.

To reward queries that pinpoint the search intent of the user, query level attention mechanism is applied.

$$u_i = tanh(\boldsymbol{W}_s \boldsymbol{h}_i + b_s). \tag{10}$$

$$\alpha_i = \frac{\exp(u_i^T u_s)}{\sum_i \exp(u_i^T u_s)}. \tag{11}$$

$$\boldsymbol{s} = \sum_i \alpha_i \boldsymbol{h}_i. \tag{12}$$

Equations 10, 11, and 12 summarizes the attention process on query level. Finally, $\boldsymbol{s}$ gives the query sequence embedding that captures the search intent embedded in sequential queries $<Q1, ...,Q_i, ..., Q_N>$. In a nutshell, given a query sequence, we first use query encoder to derive the embedding for each query in the sequence. Then, we use query sequence encoder to further construct the embedding of the query sequence.

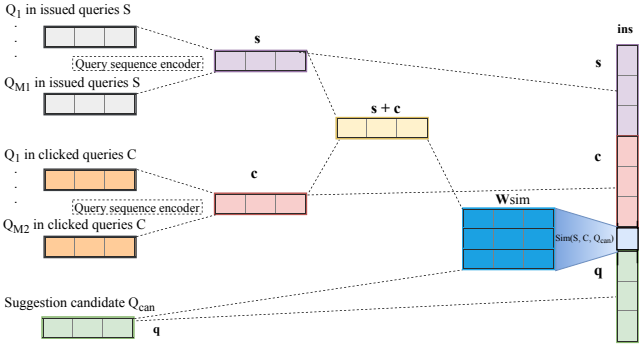

Figure 2: Training instance construction based on search sequence S, clicked sequence C, and suggestion candidate Q_{can}.

Given a training instance Ins, which is composed of a search query sequence S, a clicked query sequence C, and a suggestion candidate Q_{can}, we use the same query sequence encoder to derive the embeddings for S and C, denoted as $\boldsymbol{s}$ and $\boldsymbol{c}$, respectively, and we use a query encoder to encode the candidate Q_{can}, with its embedding denoted as $\boldsymbol{q}$. We use the concatenation of $\boldsymbol{s}$, $\boldsymbol{c}$, and $\boldsymbol{q}$, i.e., $[\boldsymbol{s}; \boldsymbol{c}; \boldsymbol{q}]$ to get the embedding **ins** to represent the instance Ins. Figure 2 summarizes the process.

4.2.2 Matching query sequences and suggestion query. Given a sequence of issued queries S, a sequence of clicked queries C, and a suggestion candidate Q_{can}, we aim to explicitly derive the matching among S, C, and Q_{can}. We follow approaches [3] and [19]. We define the matching score among $\boldsymbol{s}$, $\boldsymbol{c}$, and $\boldsymbol{q}$ as follows:

$$\text{Sim}(S, C, Q_{\text{can}}) = (\boldsymbol{s} + \boldsymbol{c})^T \boldsymbol{W}_{\text{sim}} \boldsymbol{q}, \tag{13}$$

where $\boldsymbol{W}_{\text{sim}} \in \mathbb{R}^{|\boldsymbol{s}|\times|\boldsymbol{q}|}$ is a similarity matrix. In this module, $\boldsymbol{s}+\boldsymbol{c}$ explicitly combines the search intent embedded in search queries S and clicked suggestions C. We seek a transformation of the candidate query Q_{can} that is the closest to $\boldsymbol{s}+\boldsymbol{c}$. The similarity matrix $\boldsymbol{W}_{\text{sim}}$ is a parameter of the network and is optimized during training. $\text{Sim}(S, C, Q_{\text{can}})$ is then explicitly appended to **ins** as a feature.

4.2.3 Learning and Optimization. When learning *CFAN* without adversarial examples, *CFAN* is optimized by learning only the ranker, formulated as finding a set of parameters Θ, that minimize an empirical ranking loss as shown in Equation 1 for a given set of training views. In adversarial training, we construct adversarial perturbations on training instances in order to cause a misbehavior of *CFAN* on a training view V. The misbehavior refers to the event that *CFAN* ranks unclicked suggestions ahead of clicked suggestions. To construct such adversarial perturbations, we introduce a binary classifier, which learns the labels of training instances in V. For each instance, the loss function $\text{Loss}_{\text{cls}}(\Theta)$ of the classifier focuses on reducing the binary cross-entropy [10] between the predicted probabilistic score $\hat{y}^{\Theta}_{\textbf{ins}}$ and the gold standard $y_{\textbf{ins}}$ as follows:

$$\text{Loss}_{\text{cls}}(\Theta) = -[y_{\textbf{ins}} \log(\hat{y}^{\Theta}_{\textbf{ins}}) + (1 - y_{\textbf{ins}}) \log(1 - \hat{y}^{\Theta}_{\textbf{ins}})], \quad (14)$$

where $\hat{y}^{\Theta}_{\textbf{ins}} = g_{\Theta}(\textbf{ins})$.

To increase the classification difficulty, we aim to identify a bounded perturbation $\boldsymbol{r}_{\text{adv}}$ for each **ins** such that the classification loss on $\textbf{ins} + \boldsymbol{r}_{\text{adv}}$ is as large as possible. The adversarial loss of the classifier becomes:

$$\text{Loss}_{\text{cls adv}}(\Theta) = -[y_{\textbf{ins}} \log(\hat{y}^{\Theta}_{\textbf{ins}+\boldsymbol{r}_{\text{adv}}}) + (1-y_{\textbf{ins}}) \log(1-\hat{y}^{\Theta}_{\textbf{ins}+\boldsymbol{r}_{\text{adv}}})], \quad (15)$$

where

$$\boldsymbol{r}_{\text{adv}} = \arg \min_{\boldsymbol{r}, ||\boldsymbol{r}|| \le \epsilon} [y_{\textbf{ins}} \log(\hat{y}^{\hat{\Theta}}_{\textbf{ins}+\boldsymbol{r}}) + (1 - y_{\textbf{ins}}) \log(1 - \hat{y}^{\hat{\Theta}}_{\textbf{ins}+\boldsymbol{r}})]. \quad (16)$$

$\hat{\Theta}$ is the set of current parameters defining the classifier, $\boldsymbol{r}$ is a perturbation on the input **ins**, $\hat{y}^{\hat{\Theta}}_{\textbf{ins}+\boldsymbol{r}}$ gives the predicted score of the perturbed instance based on the current set of parameters, and ϵ is a parameter to bound the perturbations.

Calculating the exact value of $\boldsymbol{r}_{\textbf{adv}}$ is not feasible, because exact minimization of Equation 16 with respect to $\boldsymbol{r}$ is intractable. To derive dynamic perturbations efficiently, we apply fast gradient sign method [8] to approximate the perturbation $\boldsymbol{r}_{\textbf{adv}}$. We linearize the cost function 16 around the current value of $\hat{\Theta}$, and obtain an optimal max-norm constrained perturbation of

$$\boldsymbol{r}_{\textbf{adv}} = -\epsilon \boldsymbol{l} / ||\boldsymbol{l}||_2, \quad (17)$$

where

$$\boldsymbol{l} = \nabla_{\textbf{ins}} \text{Loss}_{\text{cls}}(\hat{\Theta}). \quad (18)$$

Based on Equations 17 and 18, the perturbation for each instance can be calculated efficiently via back-propagation. Adversarial instances can be constructed by adding the derived perturbations $\boldsymbol{r}_{\text{adv}}$ to the corresponding instance embeddings **ins**.

To train a robust suggestion system with good ranking performance, the objective of *CFAN* combines the loss functions of both adversarial classification and ranking as follows:

$$\text{loss}(\Theta) = \sum_{\text{ins} \in V} \text{loss}_{\text{cls adv}}(\Theta) + \text{loss}_{\text{ranking}}(\Theta). \quad (19)$$

The classification component and the ranking component share the same set of parameters. The classification component dynamically generates adversarial instances, while the ranking component generates favorable rankings of query suggestions, pushing clicked suggestions to top positions and pulling unclicked suggestions to bottom positions in ranking lists.

5 EXPERIMENTS

In this section, we conduct extensive experiments and in-depth analysis to verify the performance of *CFAN* for query suggestions.

5.1 Datasets and Experimental Settings

Table 2: The statistics of queries with different context lengths in the training and test datasets.

Dataset	Context Length		
	Short (1 query)	Medium (2-3 queries)	Long (4+ queries)
Training	284,273	105,647	103,944
Test	31,577	11,998	11,566

Dataset. We use a search log collected from one of the largest search engines in the world. We pre-processed the data by eliminating non-alphanumeric characters, spelling error correction, and lower-casing. Then we segmented the search log into sessions, using a standard segmentation heuristic, i.e., intervals of at least 30 minutes idle time denotes a session boundary [12]. To partition search sessions into training and test sets, the first 90% data are utilized for training while the remaining sessions are the test data. Among the training data, 10% of sessions are randomly sampled as the validation set for parameter tuning. Finally, there are 3,684,008 training queries within 493,864 sessions and 406,063 test queries within 55,141 sessions. Furthermore, to evaluate the performance using different context lengths, the test set is partitioned into three subsets, including *Short Context* (1 query), *Medium Context* (2 to 3 queries), and *Long Context* (4 or more queries). Here the context length refers to the number of search queries in a search session. Table 2 shows the statistics of queries with different context lengths in the training and test datasets.

5.2 Evaluation Metrics

We employ Mean Reciprocal Rank (MRR) score [23] as the evaluation metric. Given a search query Q, let $Y(Q)$ be a ranked list of recommended queries determined by a suggestion method. We use $\text{rank}_1(Y(Q))$ to denote the rank of the first clicked query in $Y(Q)$. Formally,

$$\text{MRR} = \frac{1}{\text{rank}_1(Y(Q))}. \quad (20)$$

Essentially, the MRR score summarizes the ranks of the first clicked queries in the recommendation list $Y(Q)$ - A larger score indicates that the first clicked queries are ranked higher in the list.

5.3 Baselines Methods

In our evaluations, we evaluate the MRR performance of *CFAN* against the following state-of-the-art methods.

Table 3: The MRR performance of different methods in the test sets with different context lengths for the task of query suggestion.

Dataset	QVMM [9]	FBS [28]	RC [14]	MPS [7, 20]	Hybrid [1]	HRED [20]	RIN [15]	CFAN
Overall Context	0.441	0.458	0.466	0.478	0.473	0.556	0.573	**0.650**
Short Context (1 query)	0.446	0.473	0.460	0.476	0.473	0.548	0.563	**0.639**
Medium Context (2 to 3 queries)	0.430	0.438	0.477	0.483	0.473	0.564	0.591	**0.661**
Long Context (4 and more queries)	0.431	0.440	0.469	0.474	0.475	0.574	0.593	**0.682**

- **[Query-based Variable Markov Model (QVMM)]** [9] makes query suggestions by learning the probability of query transitions over search sessions with the variable memory Markov model.
- **[Feature Based Suggestion (FBS)]** [28] counts on statistical features to make suggestions. Suggestion candidates are generated based on 1) term matching between search queries and candidates and 2) query co-occurrences in search sessions. Query suggestions are made by ranking candidates based on a combination of statistical features.
- **[Reformulation-based Completion (RC)]** [14] is a non-deep learning based method exploiting query suggestions. 43 reformulation based features are proposed to capture user reformulation behaviors over search sessions with LambdaMART [4].
- **[Most Popular Suggestion (MPS)]** [7, 20] is a maximum likelihood method, which relies on "wisdom of the crowd". It ranks queries by the co-occurrence to the last query in the search sequence.
- **[Hybrid Suggestion (Hybrid)]** [1] considers both the context information and the popularity by ranking candidate queries based on a linear combination between the popularity and the similarity to recent search queries.
- **[Hierarchical Recurrent Encoder-Decoder (HRED)]** [20] is a deep learning based query suggestion method. HRED constructs a hierarchical encoder-decoder structure to model the sequential and hierarchical dependencies across terms and queries to make query suggestions.
- **[Reformulation inference network (RIN)]** [15] is the most state-of-the-art query suggestion method. Query reformulations between consecutive queries in search sessions are explicitly modeled in RIN to make suggestions.

5.4 Experimental Results

Table 3 shows the MRR performance of different methods over various context lengths. QVMM, a variable-memory Markov model, achieves the worst MRR performance among all the query suggestion methods. QVMM counts on query dependencies in search sessions to make query suggestions. The sparsity of search sessions leads to the poor ranking performance of QVMM. Feature based method (FBS) first selects candidate queries based on prefix matching and query dependencies. Then, candidate queries are ranked based on statistical popularity features. Compared with QVMM, FBS improves the overall MRR from 0.441 to 0.458. The reformulation-based completion method (RC) outperforms both QVMM and FBS. Its overall MRR score reaches 0.466. RC relies on generalized query dependency rules to make query suggestions. The improvements against QVMM and FBS demonstrate that generalized reformulation rules are more informative and powerful than simple query dependencies in search sessions. The popularity-based baseline method (MPS) slightly outperforms RC. The overall MRR score further increases from 0.466 to 0.478. It indicates that the query popularity plays an important role in driving users to click the suggestions. This is mainly because when query suggestions are highly related to the search query, users tend to click those popular and trending ones as follow-up search queries to explore. The hybrid suggestion method (Hybrid) performs slightly worse than MPS. The overall MRR score decreases from 0.478 to 0.473. This can be explained by the fact that the term-level similarities between suggestion candidates and search queries in Hybrid are not capable of capturing the their relationship in semantic, and thus fail to rank suggestion candidates properly when most suggestion candidates are similar to search queries in term levels.

Neural network based methods HRED, RIN, and *CFAN* outperform non-deep learning based methods by a large margin in general. HRED achieves MRR scores of 0.556, 0.548, 0.564, and 0.574 on overall, short, medium, and long contexts, respectively. RIN achieves MRR score of 0.573 on average. Among all the methods, *CFAN* achieves the best MRR performance. The MRR scores reach 0.650, 0.639, 0.661, and 0.682 on overall, short, medium, and long search contexts, respectively. In particular, *CFAN* achieves excellent ranking performance when the context information is rich. Rich context contains more user search queries and click feedback, which help deduce the query ambiguity and thus contribute to the ranking performance.

6 CONCLUSION AND FUTURE WORK

In this paper we investigated the problem of feedback-aware query suggestion. A novel neural network based method, *CFAN*, is proposed, which models both user search sequence and click sequence.In addition, to improve the robustness of the query suggestion system, adversarial examples are constructed via a classifier. *CFAN* is optimized by training the adversarial classifier and a ranker simultaneously. Extensive experiments demonstrate that our proposed model significantly outperforms the baseline methods for the task of query suggestion. The improvements are consistent across different datasets of various context lengths.

ACKNOWLEDGMENTS

This project was supported by NIH U01HG008488. We also appreciate the gift funds from NEC Lab and Visa.

REFERENCES

[1] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*. 107–116.

[2] Ranieri Baraglia, Carlos Castillo, Debora Donato, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri. 2009. Aging effects on query flow graphs for query suggestion. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*. 1947–1950.

[3] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open Question Answering with Weakly Supervised Embedding Models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*. 165–180.

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

[5] 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, Las Vegas, Nevada, USA, August 24-27, 2008*. 875–883.

[6] Zhiqian Chen, Xuchao Zhang, Arnold P. Boedihardjo, Jing Dai, and Chang-Tien Lu. 2017. Multimodal Storytelling via Generative Adversarial Imitation Learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 3967–3973.

[7] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. 1747–1756.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2014). arXiv:1412.6572

[9] Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web Query Recommendation via Sequential Query Prediction. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. 1443–1454.

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

[11] Chao Huang, Junbo Zhang, Yu Zheng, and Nitesh V Chawla. 2018. DeepCrime: Attentive Hierarchical Recurrent Networks for Crime Prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1423–1432.

[12] Bernard J. Jansen, Amanda Spink, Chris Blakely, and Sherry Koshman. 2007. Defining a session on Web search engines. *JASIST* 58, 6 (2007), 862–871.

[13] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. 2000. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.* 36, 2 (2000), 207–227.

[14] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*. 445–454.

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

[16] Ruirui Li, Ben Kao, Bin Bi, Reynold Cheng, and Eric Lo. 2012. DQR: a probabilistic approach to diversified query recommendation. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*. 16–25.

[17] Qiaozhu Mei, Dengyong Zhou, and Kenneth Ward Church. 2008. Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*. 469–478.

[18] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing* 45, 11 (1997), 2673–2681.

[19] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 373–382.

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

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199 (2013). arXiv:1312.6199

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 6000–6010.

[23] Ellen M. Voorhees. 1999. The TREC-8 Question Answering Track Report. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*.

[24] Ji-Rong Wen, Jian-Yun Nie, and HongJiang Zhang. 2001. Clustering user queries of a search engine. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*. 162–168.

[25] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query Suggestion with Feedback Memory Network. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 1563–1571.

[26] Xian Wu, Yuxiao Dong, Chao Huang, Jian Xu, Dong Wang, and Nitesh V Chawla. 2017. Uapd: Predicting urban anomalies from spatial-temporal data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 622–638.

[27] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, Louis Faust, and Nitesh V Chawla. 2018. Restful: Resolution-aware forecasting of behavioral time series data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1073–1082.

[28] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly Jr., Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, and Yi Chang. 2016. Ranking Relevance in Yahoo Search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 323–332.