Query Suggestion with Feedback Memory Network

Bin Wu*
Tsinghua University
wub16@mails.tsinghua.edu.cn

Chenyan Xiong*
Carnegie Mellon University
cx@cs.cmu.edu

Maosong Sun†
Tsinghua University
sms@mail.tsinghua.edu.cn

Zhiyuan Liu
Tsinghua University
liuzy@tsinghua.edu.cn

ABSTRACT

This paper presents Feedback Memory Network (FMN) which models user interactions with the search engine for query suggestion. Besides modeling the queries issued by the user, FMN also considers user feedback on the search results. It converts user browsing and click actions to the attention over the top-ranked documents and combines them into the feedback memories of the query, thus better models the underlying information needs. The feedback memories and the query sequence are then combined to suggest queries by the sequence-to-sequence neural network. Modeling user feedback makes it possible to suggest diverse queries for the same query sequence, if users have preferred different search results that indicate different information needs. Our experiments on the search log from a Chinese commercial search engine showed the stable and robust advantages of FMN. Especially when the feedback is richer or more informative, FMN provides more diverse and accurate suggestions, which is exceptionally helpful for ambiguous sessions where more information is required to infer the search intents.

KEYWORDS

Query Suggestion, Feedback Memory Network, User Modeling

1 INTRODUCTION

In modern information retrieval, a session of multiple queries is often required to complete a search task: precisely expressing the information need in a short ad hoc query sometimes can be tricky; the search engine may fail to provide relevant search results for the query; the user may decide to further explore the topic after browsing the initial search results. Query suggestion techniques, which provide query auto-completion, refinements, and related queries, have been widely adopted by search engines to facilitate this information seeking process and improve user satisfaction [2].

A successful query suggestion depends on modeling the user's information needs accurately. The information needs are reflected by the user's interactions with the search engine in the session: the *query sequence* she issued, and the *feedback* she provided on the search results. Previous context-aware query suggestions have modeled the query sequence efficiently, but the user feedbacks are merely treated as a secondary resource to help model the query sequence or even overlooked [8, 25].

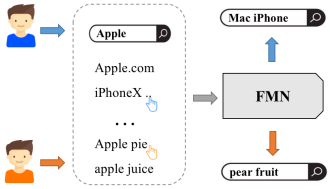


Figure 1: Information needs reflexed by user feedbacks

However, user feedbacks sometimes are necessary for search engines to infer the precise search intent behind queries. For example, as shown in Figure 1, the query "Apple" itself is ambiguous, but the user's preference on 'Apple.com' or 'Apple Juice' reflexes the search intent more precisely.

This paper presents a neural query suggestion method that models both the query sequence and the user feedback in the session, named as **F**eedback **M**emory **N**etwork (FMN). FMN embeds the search results of a query using a memory network, which converts the contents of ranked documents to distributed representations by recurrent neural networks, and calculates the attention over these documents according to the similarity between the query and those documents and also user's preference on those documents. FMN then produces the 'feedback memories' for the query by combining the content embeddings via their attention scores. In the query suggestion task, the feedback memories of queries in a session is easily integrated to a sequence-to-sequence model to produce *feedback-aware* query suggestions.

FMN is trained end-to-end using user behaviors in search logs. Given the correct query suggestions, FMN learns the query sequence model and the feedback model simultaneously. The feedback model helps distinguish the different query suggestions following the same input query sequence but with different click patterns, which would confuse the sequence-to-sequence model without feedback awareness. It also propagates the training signals across different sessions with shared search results, reducing the sparsity of the query sequence. The *feedback-awareness* introduced by FMN thus influences not only the predicting behavior of the query suggestion model, but also the learning of the model itself.

*Bin Wu and Chenyan Xiong contributed equally to this work.
†corresponding author

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186068

Our experiments on the search log from Sogou, a major Chinese commercial search engine, demonstrated FMN's robust effectiveness. Stable improvements have been observed over an unsupervised method, feature-based methods, and state-of-the-art neural methods that do not consider user feedbacks. FMN's advantages are more significant in more extreme scenarios: On sessions that contain too little information or more noisy signals, the additional feedback signals make FMN's performance more stable; on more ambivalent queries where context-aware query suggestion systems may get confused, FMN are more accurate because its feedback awareness helps locate more fine-grained search intents. Our analyses further revealed that the successful modeling of the feedback signal is the source of FMN's effectiveness: FMN produces more accurate suggestions when more feedback signals are made available, or the feedback signal is more informative.

The next section discusses the related work. The architecture of FMN and its application in query suggestion are in Section 3. Experiment settings and evaluation results are presented in Section 4 and 5. The last section concludes and discusses future work.

2 RELATED WORK

Query suggestion systems utilize the 'wisdom of crowds' to suggest semantically related queries for the input session. The semantic relatedness can be modeled by the Query Flow Graph which connects queries by their session co-occurrences [3]. It can also be described by the similarities between queries' search results [3, 28]. The query-click bipartite graph is another widely studied resource to connect queries through their shared clicks [1], for example, the relatedness can be described by the distance (hit time) in the bipartite graph [18].

A lot of techniques have been developed to address the sparsity of the 'wisdom of crowds'—a major challenge in query suggestion. The Term Query Graph enriches the Query Flow Graph with term nodes; the connections between query nodes and term nodes smooth the query flow and help find suggestions for tail queries [4]. The query-click graph and the Query Flow Graph can be united to combine the strengths of both sides [14]. The sparse signals in query suggestion can also be smoothed by clustering queries using search results [2] or the Query Flow Graph [22], and sharing information within clusters. Another line of research is to build additional connections between queries using external semantic resources, for example, templates generated from WordNet [27], shared entity annotations [6], and knowledge graph relations [13].

Cao et al. proposed the context-aware query suggestion framework [8] which considers the whole query sequence in the session, instead of only the last query. They used query clusters to build a concept sequence suffix tree, for efficient and effective context-aware query suggestions. The query sequence can also be modeled by the Mixture Variable Memory Markov Model [12]. Context-aware query suggestion considers more user actions in the session and thus better models the information needs. Hence, the idea is also effective in query classification [7] and ranking [29].

The rich signals developed in previous research have also been combined for query suggestion by machine learning techniques. For example, finding the right query substations or rewritings was considered as a classification problem [16]. Ozertem et al. [21] developed a ranking framework that learns to suggest queries directly from user's search behaviors in the search log. It utilizes the large-scale search logs and avoids the requirement of human labels. Supervised suggestion systems are in general more accurate than unsupervised ones while also being more flexible. Their suggested results can also improve diversified and personalized search [23, 24].

The sparse signals and large-scale training data make query suggestion a natural fit for deep learning approaches. Sordoni et al. [25] developed a hierarchical encoder-decoder model (HRED) for context-aware query suggestion. The encoder first uses a two-level recurrent neural network (RNN) to encode the query words to the query embedding and then the query sequence to the session embedding. It then decodes the session embedding to target suggestions. HRED avoids sparsity using smoothly distributed representations and better utilizes large-scale training data available in search logs. It achieves better accuracy than feature-based systems. A more recent work upgraded HRED's sequence-to-sequence model with the attention and coping mechanism to model the varying query importances and repeating terms in sessions [11].

This paper introduces memory network to model user feedbacks. Memory network has provided an effective way to incorporate external information into neural models [26]. A memory cell in the memory network can be considered as a key-value pair: the key generates the attention weight on the memory cell, and the value is the external information to incorporate [20]. Memory networks have been successfully adopted in many tasks, for example, reading comprehension [20] and task-oriented dialog system [5].

3 FEEDBACK MEMORY NETWORK

This section first describes the architecture of Feedback Memory Network (FMN), which models the user feedbacks on the search results and produces *feedback memories* for the corresponding query. Then it discusses how FMN is incorporated in a query suggestion system and makes it *feedback-aware*.

3.1 Model Architecture

User's preference on search results reflects more fine-grained information needs. It has been used as a static resource to infer document's relevance [10] and to train ranking models [15, 30]. FMN models the feedback signals more dynamically with neural networks. As shown in Figure 2, FMN takes the user's interactions with the search engine as inputs and converts them to distributed representations. The distributed representations are the *feedback memories* of the query and encode the information needs reflected by user's preferences, for example, the preferences on 'iPhone' or 'Apple pie' for the query 'apple'.

Given a query q, its search results $D = \{d_1, ...d_i..., d_n\}$, and the clicked positions $C = \{p|\text{User clicked on } d_p.\}$, FMN considers the clicked documents as positive feedback documents D^+, and the skipped documents as negative D^-:

$$D^+ = \{d_p | p \in C\},$$
$$D^- = \{d_p | p <= \max(C) + 1, p \notin C\}.$$

It uses the cascade assumption [10]: The user prefers documents she clicked over those she skipped that are ranked higher or one

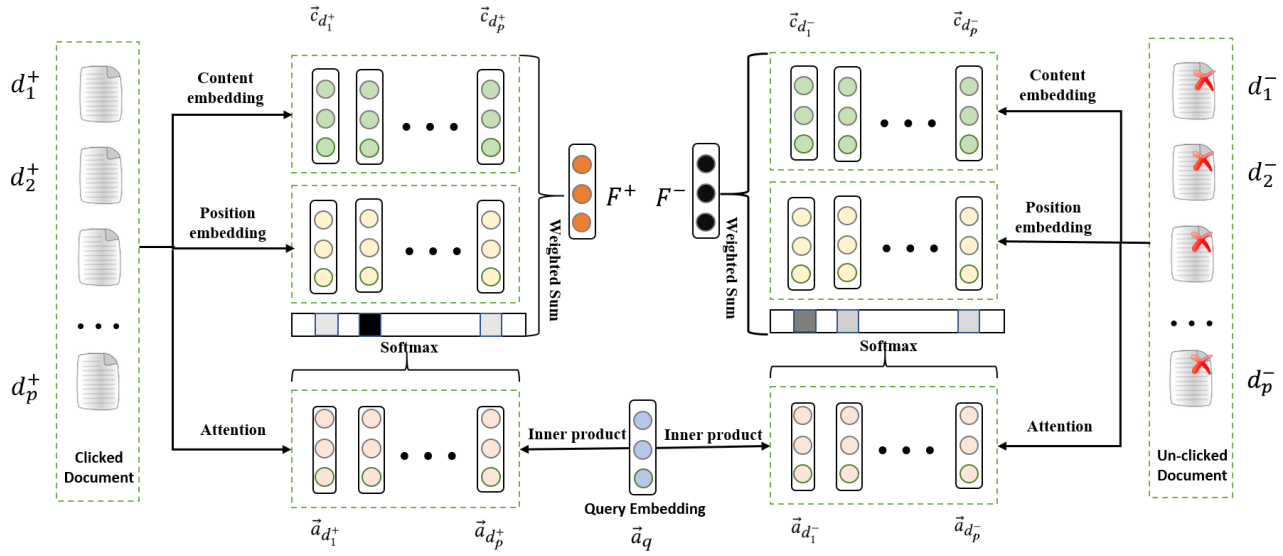


Figure 2: The architecture of Feedback Memory Network. The clicked (left) and skipped (right) search results are encoded into the positive and negative feedback memories by FMN's content encoding and position encoding. They are combined by the attention mechanism to produce the positive (F^+) and negative (F^-) feedback memories.

position lower than clicked ones, if no search result got clicked, it indicates the user is not satisfied with the first search result: $D^+ = \emptyset$ and $D^- = \{d_1\}$.

FMN encodes D^+ and D^- to the positive feedback memory F^+ and the negative feedback memory F^-, two continuous vectors representing user's preferences. It is conducted by three components: the document content encoding, the position encoding, and the attention mechanism that combines them.

Document Content Encoding: In FMN, a document's content is a sequence of its words $d_p = \{w_1^p, ...w_j^p ..., w_{|d_p|}^p\}$. As in standard sequence-to-sequence (seq2seq) learning, FMN embeds the document's words and uses a recurrent neural network (RNN) to encode it to a continuous vector $\vec{c}_{d_p}$.

$$\vec{w}_j = \text{Emb}_c(w_j^p), \tag{1}$$

$$\vec{c}_{d_p} = \text{GRU}_c(\vec{w}_1^p, ..., \vec{w}_j^p ..., \vec{w}_{|d_p|}^p). \tag{2}$$

Emb_c is the word embedding matrix for document contents. $|\text{Emb}_c| = V \times l$, where V is the size of the vocabulary, and l is the embedding dimension. GRU_c is the GRU model, a widely used RNN in seq2seq learning. $\vec{c}_{d_p}$ is the output vector (the last hidden state) of the GRU. The contents of all documents in D^+ and D^- are encoded by the same Emb_c and GRU_c.

Position Encoding: The ranking position of a document in the search results conveys the search engine's judgment about the document's relevancy. It also influences user's perception of the documents. To cover its effect, FMN includes the ranking position of a document as the position embedding:

$$\vec{pos}_p = \text{Emb}_{pos}(p). \tag{3}$$

Emb_{pos} is the position embedding matrix to be learned. Each of its rows is the embedding of a position (p).

Attention Mechanism: FMN uses an attention mechanism to weight-combine document content embedding and position embedding to feedback memories. It captures the importances of documents in D^+ and D^- when inferring the search intent. For example, if d_p corresponds to a rare intent of the query, clicking it is more informative; if d_p is a navigational result, skipping it indicates more unexpected intent.

Specifically, given the query q, positive documents D^+ and negative documents D^-, FMN learns the attention scores A^+ on D^+ and A^- on D^- from the interactions between the query and the documents.

The query $q = \{w_1^q, ...w_j^q, ..., w_{|q|}^q\}$ is encoded to an attention vector $\vec{a}_q$:

$$\vec{w}_j^q = \text{Emb}_q(w_j^q), \tag{4}$$

$$\vec{a}_q = \text{GRU}_q(\vec{w}_1^q, ..., \vec{w}_j^q ..., \vec{w}_{|q|}^q), \tag{5}$$

where Emb_q and GRU_q are the query embedding matrix and GRU models of the attention mechanism.

The documents d_p is encoded in the same way with another set of document attention parameters:

$$\vec{w}_j^p = \text{Emb}_{a_d}(w_j^p), \tag{6}$$

$$\vec{a}_{d_p} = \text{GRU}_{a_d}(\vec{w}_1^p, ..., \vec{w}_j^p ..., \vec{w}_{|d_p|}^p), \tag{7}$$

The attention weights of the query q on the documents in D^+ and D^- are the normalized dot products of their attention embedding

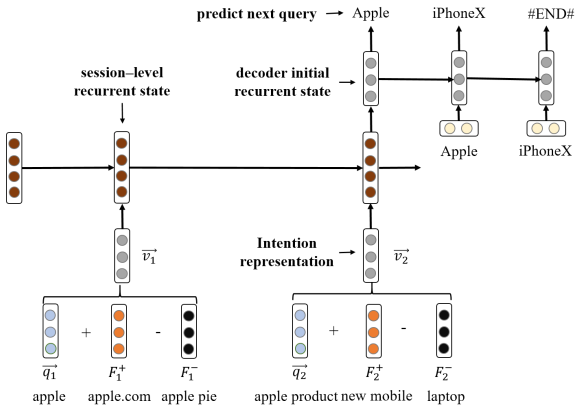


Figure 3: Feedback-Aware Query Suggestion. The feedback memories F^+ and F^- introduce the feedback signals to the encoder-decoder neural network.

to the query's:

$$A_p = \vec{a}_q^T \vec{a}_{d_p}, \tag{8}$$

$$A^+ = \text{softmax}\{A_p | d_p \in D^+\}, \tag{9}$$

$$A^- = \text{softmax}\{A_p | d_p \in D^-\}. \tag{10}$$

Note that the attention scores are normalized separately in D^+ and D^-; the user feedback signal is covered by the belonging of documents in the positive and negative sets.

Feedback Memories: The attention scores, document embedding, and position embedding together produce the positive and negative feedback memories for the query:

$$F^+ = \sum_{d_p \in D^+} A_p^+ (M(\vec{c}_{d_p} || \vec{pos}_p) + b_F), \tag{11}$$

$$F^- = \sum_{d_p \in D^-} A_p^- (M(\vec{c}_{d_p} || \vec{pos}_p) + b_F). \tag{12}$$

The position embedding and the content embedding are concatenated (||) and then projected by the to-be-learned matrix M and bias b_F. F^+ and F^- are the output vectors of FMN, representing the positive and negative preferences reflected by user feedbacks. Besides the projections, FMN's parameters include the embeddings for document content Emb_c, the query attention Emb_q, and the document attention Emb_{a_d}, as long as the corresponding GRU unites: GRU_c, GRU_q, and GRU_{a_d}.

3.2 Feedback-Aware Query Suggestion

FMN's feedback memories F^+ and F^- encode the information needs reflected by user's preferences. It can be integrated as the external memories of the query in neural query suggestion systems and then trained end-to-end using back-propagation. The integrated query suggestions are *feedback-aware*: Both query sequences and user feedbacks are considered; different queries can be suggested for the same query sequence if the click patterns are different.

This work chooses a previous state-of-the-art neural model, HRED [25], to integrate with FMN. HRED is a sequence-to-sequence model. It first encodes the query sequence $S = \{q_1, ...q_k, .., q_K\}$ to a hidden vector, and then decodes the candidate query suggestions from the vector. FMN is plugged in to enrich context-aware query representations for the encoder, using the architecture shown in Figure 3. The rest of this section describes the integration architecture, the candidate query suggestion scoring process, and the model training using search logs.

Encoding with FMN: The query sequence and feedback memories are encoded by a two-level encoder:

The first level encodes each query q_k by the standard seq2seq model:

$$q_k \xrightarrow[\text{Emb}_q]{\text{GRU}_q} \vec{q}_k, \tag{13}$$

which is similar to Equation 4 and 5 and has the same parameters Emb_q and GRU_q. The resulted query content embedding $\vec{q}_k$ conveys the information needs reflected by the query string.

The second level encodes the session's query contents and feedback memories to the session embedding $\vec{S}$. It first combines each query's content embedding with its feedback memories:

$$\vec{v}_{q_k} = \vec{q}_k + F_{q_k}^+ - F_{q_k}^-. \tag{14}$$

$\vec{v}_{q_k}$ is the query 'intent' representation and contains signals from the query content and the user preferences on the search results.

The intent representations of the query sequence are combined by the session level GRU_s:

$$\vec{S} = \text{GRU}_s(\vec{v}_{q_1}, ...\vec{v}_{q_k}, ...\vec{v}_{q_K}). \tag{15}$$

The session embedding $\vec{S}$ includes information from the query sequence and the user feedbacks.

Decoding: The decoder decodes the session embedding $\vec{S}$ to the target query suggestion. This part is the same with HRED [25], except that $\vec{S}$ is enriched with the feedback-awareness.

HRED first transforms $\vec{S}$ to the initial state of the decoder:

$$h_0 = \tanh(D\vec{S} + b_0). \tag{16}$$

$|D| = |h_0| \times |\vec{S}|$ is the projection and b_0 is the bias. tanh() is the activation function.

The target query suggestion $q_s = \{w_1^s, .., w_j^s, ..., w_{|q_s|}^s\}$ is decoded by another GRU:

$$h_j = \text{GRU}_{dec}(h_{j-1}, w_{j-1}^s), \tag{17}$$

and the probability of generating the next word w_j is:

$$p(w_j | w_1 : w_{j-1}, \vec{S}) = \text{softmax}(\vec{w}_j^T f(h_{j-1}, \vec{w}_{j-1})), \tag{18}$$

$$f(h_{j-1}, \vec{w}_{j-1}) = Hh_{j-1} + E\vec{w}_{j-1} + b_{prob}, \tag{19}$$

$$\vec{w}_0 = \vec{0}. \tag{20}$$

f is a dense layer with parameters H, E, b_{prob}: $|H| = |\vec{w}_j||h_0|$, $|E| = |\vec{w}_j|^2$ and $b_{prob} = |\vec{w}_j|$. The softmax is taken over all possible candidate words, for example, those appeared in candidate queries or the entire vocabulary. $\vec{0}$ is an all-zero vector.

Scoring Candidate Suggestions: Instead of directly generating a query, a more conservative choice is to rank candidate query suggestions using the decoder. The score of a candidate query q_s is

Table 1: Statistics of the dataset used in the experiments.

	Training	Development	Testing
Number of Sessions	7,978,441	3,989,220	36,519
Number of Queries	27,191,564	13,551,903	117,225

the probability of it being decoded given the session embedding:

$$s(q_s) = \prod_j p(w_j | w_1 : w_{j-1}, \vec{S}). \quad (21)$$

The score can be integrated into a feature-based query suggestion system, as in HRED [25].

Model Training: The whole model, including the feedback memory network and the hierarchical encoder-decoder, are trained end-to-end using sessions in the search log.

Given the queries, search results, and user feedbacks in a training session, the last query of the session is treated as the correct query suggestion q_s [21]. The training is conducted by maximizing the likelihood of q_s given the other part of the training session:

$$l = \sum_{w_j \in q_s} \log p(w_j | w_1 : w_{j-1}, \vec{S}).$$

Standard back propagation is used to send the gradients from the likelihood to the decoder, the encoder, and then to FMN. The query sequence model and the memory networks are optimized jointly for better query suggestion accuracy.

4 EXPERIMENT

This section discusses the dataset, baselines, implementation details, and evaluation metrics in our experiments.

Dataset: Our experiments are conducted on a large scale Chinese search log from Sogou, a Chinese commercial search engine. The search log includes queries, the titles and URL's of displayed documents, and clicks. Standard 30-minutes gap is used to split the queries into sessions. As a query suggestion task, only sessions with more than one query are used; the last query in a session is treated as the correct suggestion [21, 25]. The content of a document is its title, because we are not able to obtain the body text for enough documents by crawling or from the Sogou-T corpus [9]. All the queries and document titles are in Chinese. We segmented them using the THULAC open source software [17]. The first 90,000 most frequent Chinese words are used, the rest are replaced by UNK. After the segmentation, everything is treated the same as in English.

The sessions are randomly split into three parts: training (60%), development (30%), and testing (10%). The training fold trains the neural baselines and our method; the development fold trains the supervised feature-based systems; the testing fold evaluates all methods [25]. The statistics of the three folds are listed in Table 1.

The re-ranking setting in prior work [25] is used in our experiments, in which the query suggestion systems are evaluated by their ability to re-rank the candidate suggestions. The candidates are the top-20 most frequent follow-ups in the search log for the input query sequence. We chose this conservative setting because it performs better than the generation-based setting [25] and to focus on evaluating the effectiveness of FMN in modeling user feedbacks.

Table 2: Parameters to learn in the neural methods. The USE column marks the models that use the corresponding parameters: H refers to HRED, P refers to PRFMN, and F refers to FMN. The bracketed numbers (d_1, d_2, d_3) are the input size, hidden state size, and the number of layers of GRU's, or the dimension of the matrix parameters. The vocabulary is the $90k$ most frequent words in the search log.

Parameter	Dimension	USE	Description
Emb_q	(90k, 256)	HPF	Query Content
GRU_q	(256,256,1)	HPF	Query Content
GRU_s	(256,512,1)	HPF	Session Encoder
GRU_{dec}	(256, 512, 1)	HPF	Decoder
D	(512,256)	HPF	Decoder Projection
b_0	256	HPF	Decoder Bias
H	(256, 512)	HPF	Decoder Probability
E	(256, 256)	HPF	Decoder Probability
b_{prob}	256	HPF	Decoder Probability
Emb_{a_d}	(90k, 256)	PF	Document Attention
GRU_{a_d}	(256, 256, 1)	PF	Document Attention
Emb_{c_d}	(90k, 256)	PF	Document Content
GRU_{c_d}	(256, 256, 1)	PF	Document Content
Emb_P	(15, 4)	F	Position Embedding
M	(256,260)	F	Feedback Projection
b_F	256	F	Feedback Bias

Baselines: The baselines compared include frequency-based, feature-based, and neural-based methods.

Frequency-based: The ADJ baseline ranks the candidate queries solely by their frequencies following the original query sequence in the search log [25].

Feature-based: We implemented the feature-based baseline method in previous work [25]. It is a learning-to-rank model using state-of-the-art query suggestion features. One can consider it as the combination of conventional query suggestion systems. This paper refers to it as LeToR. It is also the base query suggestion system—all the other methods except ADJ are evaluated by their effectiveness when serving as additional ranking features in LeToR [25].

For more fair comparisons, we implemented another two groups of features to model the feedback information. The first is PRF-Feature, which models the connections between the candidate query and the displayed documents in the session. The second is Feedback-Feature, which models the connections between the candidate query and the clicked documents. Three features are extracted for either of them: the query-document co-occurrence in the search log, their contents' Levenshtein (edit) distance, and the average embedding distance between their word embeddings from word2vec [19].

Neural-based: The main neural baseline is HRED [25], the previous state-of-the-art and the query sequence model used with FMN. We also compare with a degraded version of FMN which treats all displayed results as positive documents (Pseudo Relevance Feedback), named as PRFMN.

Implementation Details: LambdaMart is the ranking model of LeToR, PRF-Feature, and Feedback-Feature. They are trained on

Table 3: Overall accuracy of the query suggestions systems. All methods starting with "+" are evaluated as additional features in LeToR. MISS@K are the fraction of sessions whose correct suggestions are not ranked in top K by the corresponding method, the lower the better. Relative performances over LeToR are shown in percentages. Win/Tie/Loss are the number of sessions a method improved, did not change, or hurt, compared with LeToR. The superscripts[1,2,3,4,5] mark the statistical significant improvements over LeToR[1], +PRF Feature[2], +Feedback Feature[3], +HRED[4] and +PRFMN[5]. Best results of each metric are marked Bold.

Method	MRR		MISS@3		MISS@5		Win/Tie/Loss
ADJ	0.4928	-.5.43%	0.3165	10.7%	0.1484	1.69%	7,232/18,038/11,249
LeToR	0.5211	–	0.2859	–	0.1361	–	–/–/–
+PRF Feature	0.5285^{1}	1.42%	0.2799^{1}	-2.10%	0.1295^{1}	-4.85%	9,871/17,875/8,773
+Feedback Feature	$0.534^{1,2}$	2.48%	$0.2675^{1,2}$	-6.43%	$0.1121^{1,2}$	-17.63%	11,869/14,607/10,043
+HRED	$0.537^{1,2}$	3.05%	$0.2367^{1,2,3}$	-17.20%	$0.1175^{1,2,3}$	-13.67%	12,961/12,709/10,849
+PRFMN	$0.5358^{1,2}$	2.82%	$0.2628^{1,2}$	-8.08%	$0.1207^{1,2,3}$	-11.32%	12,343/13,804/10,372
+FMN	$\mathbf{0.5812}^{1,2,3,4,5}$	11.53%	$\mathbf{0.1921}^{1,2,3,4,5}$	-32.80%	$\mathbf{0.1085}^{1,2,3,4,5}$	-20.28%	13,146/17,528/5,845

the development fold [25]. The neural models are first trained on the training fold, and then combined with LeToR by LambdaMart using the development fold. Their parameters include the word embeddings, GRU, and the projection weights used in their components, as listed in Table 2.

All neural methods are trained using the Adadelta optimizer, with mini-batch size 64, learning rate 0.01, and anneals every 25 epochs by $\eta/2$ until 100 epochs were reached. The weights were initialized randomly from a Gaussian distribution with zero mean and $\sigma = 0.1$. On a common GPU machine and our PyTorch based implementation, FMN takes about two and half days to converge and HRED takes about one day and a half.

Evaluation Metrics: Our main evaluation metric is MRR, the standard metric in query suggestion. We also include MISS@3,5 which is the fraction of those test sessions whose correct suggestions are not ranked in top3, 5. Statistical significances are tested using the permutation (Fisher's Randomization) test with $p < 0.05$.

5 EVALUATION RESULTS

This section first evaluates the overall accuracy of FMN and its performances in difficult scenarios. Then it analyzes the effectiveness of the feedback signals in FMN.

5.1 Overall Accuracy

The overall evaluation results are in Table 3. The feature-based system, LeToR, performs about 5% better than the frequency-based ADJ, similar to the relative improvement in prior research [25]. The two additional feature groups, +PRF Feature and +Feedback Feature, provided some additional gains but in rather small margins. HRED significantly outperforms LeToR and +PRF Feature, and performs better than +Feedback Feature on earlier positions. It benefits from the large amount of training data available in the search log, and is able to learn different semantic relations that might not be covered by the manual features [25].

FMN outperforms all baselines on all metrics. It provides more improvements than all other features: With 17 manually extracted features, LeToR improves ADJ by 6%, while FMN itself improved LeToR by more than 11%. The improvements are also stable. Compared with the base system LeToR, only 16% sessions are hurt and more than twice are improved, both the best among all the methods. The feedback awareness greatly improves the neural query suggestion. Compared with HRED, which uses the same query sequence modeling but without feedback-awareness, FMN improves the MRR by 8%, and reduces the missed hit in the top 3 by 19%. Knowing user's preference is essential to utilize the search results. PRFMN uses the search results and encodes them by memory networks, but treats all search results as (pseudo) relevant. The PRF signals are too noisy to be useful: PRFMN performs worse than HRED which completely ignores the search results.

FMN models user preferences more effectively. It produces much more accurate suggestions than Feedback Feature. It is not easy to design rich features to model the feedback information. On the contrary, FMN's distributed representations and neural networks learn user's preferences from the large scale search log directly and effectively.

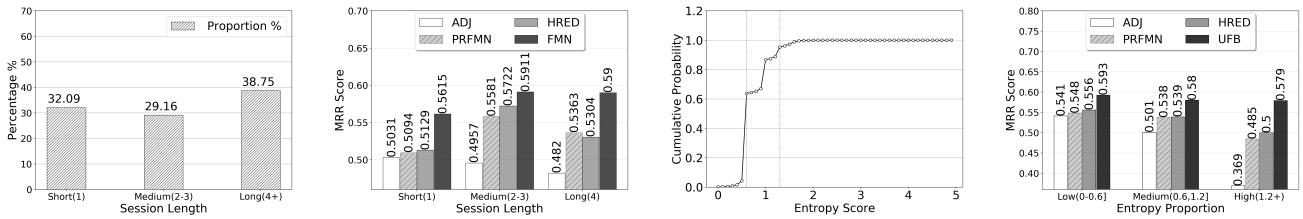
5.2 Robustness

On the queries where the 'wisdom of the crowd' is reliable, all query suggestion systems can provide reasonably accurate results. What makes a query suggestion system more desirable in real production system is its ability to handle more difficult scenarios, where the 'wisdom of the crowd' is not as effective. This experiment evaluates the FMN's accuracy in two such hard scenarios.

Short and Long Sessions: In context-aware query suggestion, the varying session length—the number of original queries—imposes challenges to the suggestion system. A too short session may have too few signal to infer the information needs. A too long session, on the other hand, may include noisy queries and drifted search intents. A robust query suggestion system should be able to provide accurate suggestions for sessions at variant lengths.

We group the testing sessions by their lengths and evaluate FMN in each group. The groups are short sessions (1 original query), medium sessions (2-3 original queries), and long sessions (4+ original queries). The distribution of testing sessions at each length group and corresponding evaluation results are plotted in Figure 4a and 4b.

As sessions become longer, ADJ performs worse. The frequency-based method suffers from the noisy and sparse signals on longer



(a) Session Length Distribution (b) MRR VS. Session Length (c) Click Entropy Distribution (d) MRR VS. Click Entropy

Figure 4: Performance on difficult scenarios. Fig 4a and Fig 4b illustrate the distributions of testing sessions at different length, and the systems' accuracies on each group. Fig 4c shows the Cumulative Distribution Function of query's click entropy. The three jumps in the CDF divides the click entropies into three groups. Fig 4d is the evaluation results on the sessions grouped by their last query's entropy.

sessions. On short sessions, HRED only provides slight improvements; its advantages are more on sessions with 2-3 original queries, where its two-level sequence learning model has more leverage. On the other end, when there are more than 3 input queries, HRED's vanilla sequence-to-sequence learning may be misled by background queries, intend drift, or perhaps also the gradient vanishing problem. More advanced sequence modeling technique such as the attention mechanism may be required [11].

FMN performs the best on all three groups. Compared to HRED, the advantages of modeling user feedbacks are actually more remarkably observed in more extreme cases. On long sessions, user clicks help FMN stay on track and its effectiveness stays the same as on medium sessions. On short sessions, the additional information from user feedbacks help reflect the information needs of the sole input query. FMN is the only method that outperforms ADJ by a large margin.

Ambivalent Queries: One advantage of feedback-aware query suggestion is that the feedback signals can help infer the information needs for ambivalent queries. For example, the query 'apple' can refer to the company or the fruit; the search target can be Apple's homepage or recent products. Without additional information, the best search engines can do is to bet on the most popular intent or to diversify. But with the feedback signals, the information need becomes much more clear: the user's preference on 'Apple.com', 'apple pie', or 'iPhone' points out what she wanted.

This experiment evaluates FMN on ambivalent queries. The ambivalence of a query is described by the entropy of user clicks on its search results. A more scattered click (high entropy) indicates more ambivalent search intents, while if all users clicked on the same result, it might be an easy query for query suggestion systems.

Figure 4c plots the Cumulative Distribution Function (CDF) of the testing queries' click entropy. There are three main jumps in the CDF, which divide the click entropy into three groups: low (entropy <=0.6, 4.33% queries), medium (entropy in between 0.6-1, 83.11% queries), and high (entropy > 1.2, 12.56% queries). We grouped the testing sessions based on their last query's entropy, and evaluated the performances in each group. The results are in Figure 4d.

The frequency-based method suffered on ambivalent queries: ADJ's MRR drops 30% from low entropy to high entropy sessions. HRED's accuracy also drops, though slightly less than ADJ because HRED considers the entire sequence as a context which can disambiguate the last query. However, the query sequence alone may not be sufficient to reflect the information needs. There is still more than 10% difference on HRED's MRR's between the low and high entropy sessions. FMN's performances are very stable in the three groups. The feedback signals greatly reduce the ambiguity of the query sequence. The feedback-aware system can provide accurate query suggestions even on high entropy sessions, with which the context-aware systems is difficult to deal.

5.3 Source of Effectiveness

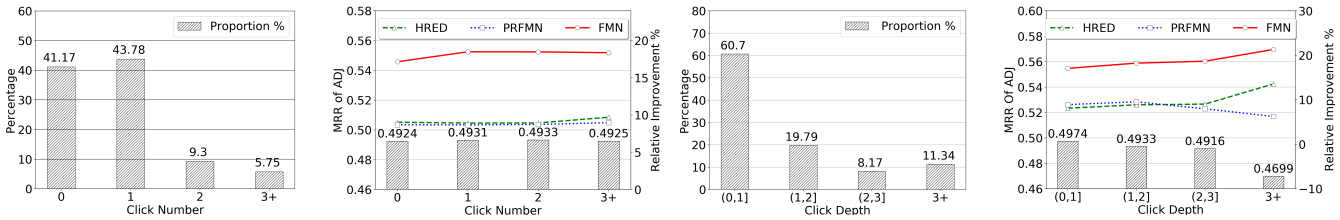
This experiment analyzes the influence of feedback signals in FMN.

Feedback Strengths: We first analyze the influence of feedback signals' strength in suggestion accuracy. We present two ways to describe the signal strength: number of clicks and the average click depths in the session.

Number of Clicks. More clicks in a session provide more feedback signals and could improve FMN's accuracy. We divide the testing sessions into four groups based on their numbers of clicks: no click, one click, two clicks, and three plus. The distribution of the four groups are shown in Figure 5a. The majority of sessions have zero or one click. Those sessions with more than two clicks are less common. The MRR of ADJ, and the relative improvements of HRED, PRFMN and FMN over ADJ in the fours groups are plotted in Figure 5b.

ADJ, HRED, and PRFMN perform similarly across the four groups as none of them uses the feedback signals. The relative improvement from FMN receives a slight jump from zero click to one click. It performs about $1-2\%$ better when relevance feedback signals are available. Note that on no click sessions, FMN still outperforms HRED by a large margin. The reason is that no click indicates that the user is unsatisfied with the first search result, which is the negative feedback signal used by FMN—it is still better than no feedback and can be effectively utilized by FMN.

Click Depths. A click on a lower ranked result implies that the user has skipped the top ranked ones. Her information need is more unexpected by the search engine. It might be a stronger feedback signal than click on the top ranked results. We divided the testing sessions by their average click depths into four groups: [0, 1], (1, 2], (2, 3], and (3, ∞). Their fractions and corresponding model performances are shown in Figure 5c and 5d.



(a) Click Number Distribution (b) MRR VS. Click Number (c) Click Depth Distribution (d) MRR VS. Click Depth

Figure 5: Performance with different amounts of feedback information. Fig 5a and 5c show the distribution of test sessions in corresponding groups. Their x-axises mark the range of each group, and y-axises are the fractions in the corresponding group. Fig 5b and 5d are the evaluation results of the query suggestion systems in each group. The absolute MRR's of ADJ are bared by their left y-axes; the relative improvements of other methods are on the right y-axes.

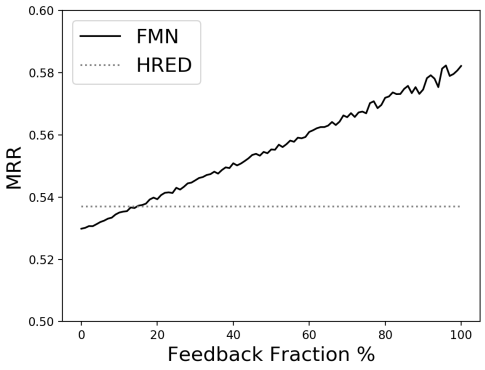


Figure 6: Performance of FMN with different amounts of user feedbacks. The x-axis is the fractions of feedbacks provided to FMN when testing; the Y-axis is MMR.

The ADJ's MRR negatively correlates with the click depth: a lower click indicates more ambivalent search intent. In comparison, FMN's relative improvement is positively correlated with the click depth. FMN has more leverage with the stronger feedback signal, which is also more useful for more ambivalent search intents as indicated by the deeper clicks.

Feedback Fraction: The second analysis studies FMN's performance with different amounts of feedback signals. When testing, we randomly discard a certain fraction of user clicks and evaluate FMN accuracy accordingly. The results are plotted in Figure 6. The x-axis is the fraction of clicks used, from none of them are used (0%) to all of them (100%). The straight line is HRED.

Figure 6 confirms that the source of FMN effectiveness is its feedback-awareness. Without feedback signal, FMN performs worse than HRED. Recall that the no click session's negative feedback signal is helpful for FMN. However, if all feedback signals are discarded, the positive and negative feedbacks are mixed, which confuses FMN. As more feedback signals are included, FMN's accuracy becomes better and better. This strong positive correlation is another evidence for the effectiveness of feedback-aware query suggestion.

Examples: Table 4 provides some examples of feedback-aware query suggestion. For the same original query, we select the sessions where different search results were clicked, and list the queries suggested by FMN. These example queries can refer to various possible information needs, but the clicked documents reflect the search intents more clearly. Without additional contexts, the same query suggestion would be produced for the original query and miss the diverse search intents, but FMN successfully leveraged the feedback signal and produced proper query suggestions.

6 CONCLUSIONS AND FUTURE WORK

This paper proposes the Feedback Memory Network (FMN) to model user feedbacks during a search. The clicked and skipped search results are more fine-grained reflections of the information needs behind the original query. FMN encodes the clicked and skipped documents as the positive and negative feedback memories to represent user's preference in the query. These feedback memories encode the feedback signals provided by the user when interacting with the search results.

This paper integrates FMN with the hierarchical sequence-to-sequence neural query suggestion model [25]. It enriches the query representations in the sequence models with the feedback memories of FMN. The feedback memory network, and the sequence-to-sequence modeling of the query sequences are trained end-to-end using the search log. It leads to a systematic data-driven approach for feedback-aware query suggestion.

Our experiments on a large scale search log from Sogou, a major Chinese search engine, demonstrated the robust advantages of FMN. Significant improvements are consistently observed over a frequency-based method, feature-based methods, and neural methods that do not consider user feedback. FMN's advantages are more remarkably observed in more difficult scenarios. On too short or too long sessions where the information needs are less clear, FMN's improvements are larger as the user preferences provide additional signals. On ambivalent queries, FMN also performs better because user preferences on the search results conveyed more fine-grained information and helped infer the information needs.

Our analyses further demonstrate the influence of the feedback signals on query suggestion. When sessions have more clicks or when the user clicked on lower ranked search results, the feedback

Table 4: Examples of FMN's query suggestions. The clicked documents reflect the variant information needs behind the same query. FMN incorporates the feedback signals and produces feedback-aware query suggestions.

Query	Clicked Document	Suggested Query
Apple	Apple serial number lookup	Apple serial number
	Apple 110 official website	Apple official website
	Apple (China) - Official Website	Apple China official website
	Apple club serial number query	Apple serial number
Lee Sedol	Man VS machine war, tencent.com	Lee Sedol man VS machine war
	Lee Sedol, Sogou Encyclopedia	Lee Sedol
	Google AI AlphaGo crack chess game - the era of science and technology	AlphaGo VS Lee Sedol on live
	Who is the best go player, Lee Sedol or Lee Chang-Ho? Chinese player Ke Jie beat Lee Sedol to win the championship	Ke Jie beat Lee Sedol
2018 FIFA world cup	Beautiful pictures of the 2018 FIFA world cup.	2018 FIFA world cup pictures
	2018 where is the FIFA World Cup held ?	2018 FIFA world cup host city
	2018 Russia FIFA world cup, Sogou Encyclopedia	2018 Russia FIFA world cup

signals are richer and more informative, and FMN performs better. The feedback awareness also improves the query sequence modeling. They help the neural model fit the search behavior better and improve the suggestion accuracy, even when some feedback signals are omitted during testing.

FMN aims to provide a general approach to model user's interactions with the search engine. In the future work, we plan to integrate feedback memory network to other query suggestion models as well in other information retrieval tasks.

7 ACKNOWLEDGEMENT

This work is supported by the National 973 Program (No.2014CB340501) and the Major Project of the National Social Science Foundation of China (No.13&ZD190). Chenyan Xiong is supported by National Science Foundation (NSF) grant IIS-1422676. We thank Sogou for providing free access to the search log. Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Ricardo Baeza-Yates and Alessandro Tiberi. 2007. Extracting Semantic Relations from Query Logs. In *Proceedings of KDD*. 76–85.
[2] Ricardo A Baeza-Yates, Carlos A Hurtado, Marcelo Mendoza, et al. 2004. Query Recommendation Using Query Logs in Search Engines. In *Proceedings of EDBT*.
[3] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The Query-flow Graph: Model and Applications. In *Proceedings of CIKM*. 609–618.
[4] Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. 2012. Efficient Query Recommendations in The Long Tail via Center-piece Subgraphs. In *Proceedings of SIGIR*. 345–354.
[5] Antoine Bordes and Jason Weston. 2016. Learning End-to-end Goal-oriented Dialog. *arXiv preprint arXiv:1605.07683* (2016).
[6] Ilaria Bordino, Gianmarco De Francisci Morales, Ingmar Weber, and Francesco Bonchi. 2013. From Machu_Picchu to Rafting the Urubamba River: Anticipating Information Needs via The Entity-query Graph. In *Proceedings of WSDM*. 275–284.
[7] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. 2009. Context-aware Query Classification. In *Proceedings of SIGIR*. 3–10.
[8] 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 KDD*. 875–883.
[9] Luo Cheng, Zheng Yukun, Liu Yiqun, Xu Jingfang, Zhang Min, and Ma Shaoping. 2017. SogouT-16: A New Web Corpus to Embrace IR Research. In *Proceedings of SIGIR*.
[10] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click Models for Web Search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7, 3 (2015), 1–115.
[11] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. *arXiv preprint arXiv:1708.03418* (2017).
[12] 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 ICDE*. 1443–1454.
[13] Zhipeng Huang, Bogdan Cautis, Reynold Cheng, and Yudian Zheng. 2016. KB-Enabled Query Recommendation for Long-Tail Queries. In *Proceedings of CIKM*. 2107–2112.
[14] Alpa Jain, Umut Ozertem, and Emre Velipasaoglu. 2011. Synthesizing High Utility Suggestions for Rare Web Search Queries. In *Proceedings of SIGIR*. 805–814.
[15] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-rank with Biased Feedback. In *Proceedings of WSDM*. 781–789.
[16] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating Query Substitutions. In *Proceedings of WWW*. 387–396.
[17] Zhongguo Li and Maosong Sun. 2009. Punctuation as Implicit Annotations for Chinese Word Segmentation. *Computational Linguistics* 35, 4 (2009), 505–512.
[18] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. 2008. Query Suggestion Using Hitting Time. In *Proceedings of CIKM*. 469–478.
[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of Words and Phrases and Their Compositionality. In *Proceedings of NIPS*.
[20] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value Memory Networks for Directly Reading Documents. In *Proceedings of EMNLP*.
[21] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to Suggest: a Machine Learning Framework for Ranking Query Suggestions. In *Proceedings of SIGIR*. 25–34.
[22] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering Query Refinements by User Intent. In *Proceedings of WWW*. 841–850.
[23] 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.
[24] Milad Shokouhi. 2013. Learning to Personalize Query Auto-completion. In *Proceedings of SIGIR*. 103–112.
[25] 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 CIKM*.
[26] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end Memory Networks. In *Proceedings of NIPS*. 2440–2448.
[27] Idan Szpektor, Aristides Gionis, and Yoelle Maarek. 2011. Improving Recommendation for Long-tail Queries via Templates. In *Proceedings of WWW*. 47–56.
[28] Hossein Vahabi, Margareta Ackerman, David Loker, Ricardo Baeza-Yates, and Alejandro Lopez-Ortiz. 2013. Orthogonal Query Recommendation. In *Proceedings of RecSys*. 33–40.
[29] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. 2010. Context-aware Ranking in Web Search. In *Proceedings of SIGIR*. 451–458.
[30] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of SIGIR*.