Improving Query Suggestion through Noise Filtering and Query Length Prediction

Liang Wu
Computer Network Information Center, Chinese Academy of Sciences
wuliang@cnic.cn

Bin Cao
Microsoft Corporation
bincao@microsoft.com

Yuanchun Zhou
Jianhui Li
Computer Network Information Center, Chinese Academy of Sciences
{zyc, lijh}@cnic.cn

ABSTRACT

Clustering-based methods are commonly used in Web search engines for query suggestion. Clustering is useful in reducing the sparseness of data. However, it also introduces noises and ignores the sequential information of query refinements in search sessions. In this paper, we propose to improve cluster based query suggestion from two perspectives: filtering out unrelated query candidates and predicting the refinement direction. We observe two major refinements behaviors. One is to simplify the original query and the other is to specify it. Both could be modeled by predicting the length (number of terms) of queries when candidates are being ranked. Two experimental results on the real query logs of a commercial search engine demonstrate the effectiveness of the proposed approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Application

Keywords

Query suggestion, web search, search log analysis

1. INTRODUCTION

Query suggestion has been widely used to improve user experiences in Web search. The suggestions are designed to assist users to reformulate their queries to meet their information needs or recommend other content they may also get interested in. A mainstream of query suggestion methods is based on clustering: Firstly, queries are clustered based on their click patterns, and suggestion candidates are selected from the clusters where the query belongs; Secondly, machine learning and data mining techniques are employed to rank the suggestion candidates. The clustering-based methods is very effective in reducing the sparsity of data. However, it also introduces some issues due to the nature of clustering algorithm. The first problem is that clustering unavoidably brings noises when building up query clusters. Since the distribution of search queries is heavy-tailed, which means that the size of generated clusters varies significantly. In some big clusters, there often exist queries on diverse topics. Suggestions on other topics may be annoying for users, especially when they are popular, e.g., given a query "download microsoft update", the top popular query in a cluster in which it appears is "microsoft downloads 7", while a better one "microsoft update center" ranks lower in the cluster. Our proposed approach is to increase the probability of the second query to be selected as a candidate. The second problem is that cluster-based method loses the sequential information hidden in clicks in a search session. The sequence of queries are normally not encoded into clusters due to the nature of clustering. Moreover, even we want to build clusters based on the sequential information of queries, the sparsity problem would be more severe.

In order to solve the first problem, we introduce a better measurement to calculate the similarity between the suggestion and the query. We collect better candidates from a cluster by take both similarity and representative into consideration. To solve the second issue, we propose to capture the sequential information by predicting the refinement direction. We observe two major refinements behaviors in the data. One is to simplify the original query which means the user want to search in a larger scope. The other is to specify it which means the user would like to be more specific on the information need. We find both case could be identified by predicting the query length, which is the number of terms, for the following refinement queries.

2. UNRELATED CANDIDATE FILTERING

The generation of Clicked URL-based clusters can be found in [1]. A cluster $C : \{(q, p)|q \in C\}$ consists of queries q and queries' popularity p. p is measured by the number of clicks that fall in the cluster. A query q consists of at least one term t. Traditionally, when a user inputs a query q_0, candidate selection only depends on the popularity p of each query q, and the content of q_0 is not exploited. In order to filter out the unrelated candidates in big clusters, a term weight tw is is first calculated for each term in a cluster: $tw = tf \times iqf$, where term frequency tf is the number of clicks of a term which fall in the cluster; $iqf = \log \frac{|C|}{|\{q \in C : t \in q\}|}$ is the inverse query frequency. tf constrains the weight of unpopular words and iqf represents the discriminating power of a term.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2577339.

Table 1: Results of taking candidates from clusters containing more than 100 queries

	P@10	P@20	NDCG@5	NDCG@20
Cluster	0.0015	0.0013	0.0066	0.0067
Filtering	0.0019	0.0018	0.0081	0.0083
Improvement	27%	38%	23%	24%

Table 2: Results of taking candidates from clusters containing more than 500 queries

	P@10	P@20	NDCG@5	NDCG@20
Cluster	0.011	0.011	0.0039	0.0039
Filtering	0.018	0.017	0.0057	0.0057
Improvement	64%	55%	46%	46%

When q_0 is given, its cosine similarity s with each candidate is produced based on the term weight tw. Candidates are then selected by $s \times p$, instead of p only. The method can be deployed efficiently to online systems as it only adds an independent weighting module.

3. QUERY LENGTH-SENSITIVE RANKING

In order to capture the sequential order of queries, we adopt the query length as a feature when ranking the candidates. Since the change of query length can often reflect how a user's information need is satisfied, e.g., when a query is not specified enough, adding more terms into the query makes it more detailed; when a detailed query still recalls inaccurate information, she might change a term to try a different way. Here we propose to use a linear regression model to predict the length of the good suggestion and features are listed below:

Length of Query: The length of the original query q_0.
Avg Length of Candidates in the Cluster: The average number of terms in queries from clusters in which q_0 appeared.
Avg Query Length of Most Clicked Query: The average number of terms of most clicked queries in clusters in which q_0 appeared.
Number of Click: The average number of clicks of q_0 in clusters.
Ratio of Click: The proportion of clicks of q_0 in clusters.
Rank in Cluster: The average quantile of q_0 in clusters.

The first three features are based on the length of queries and the rest features measure how popular q_0 is, since a more frequently clicked query may possess a higher probability to fulfill user's information need, and the popularity of a query can be useful for revealing how we should reformulate the query.

The predicted length L is then incorporated into the ranking system as a new feature. The feature is produced as illustrated in equation 1, where *candidate.length* denotes the number of terms in a suggestion candidate.

$$feature = \frac{1.0}{exp(|candidate.length - L|)} \tag{1}$$

Table 3: NDCG@10 of the proposed method and baselines

	baseline	filtering	filtering+length
NDCG@10	0.00164	0.00166	0.00198
Improvement	0.0%	1.2%	20.7%

4. EXPERIMENTS

The dataset we used here was collected from one major commercial search engine, containing over 100 million queries and over 10 million query clusters. In order to evaluate our model, we extract 178,546 queries and its frequently clicked suggestions from user sessions for training and testing, and the noises and spams are pruned by automatic techniques and annotators' manual work. Precision at top k (p@k) and Normalized Discounted Cumulative Gain at top k (NDCG@k) are adopted as our evaluation metrics. For both experiments, five fold cross validation is used.

In both experiments, commonly used features and methods are adopted as baselines, since the performance can be improved by directly adding the proposed methods to query suggestion systems as pre- and post-processing.

Table 1 and 2 illustrate the experimental results of the candidate filtering method on big clusters. In this experiment, we only extract candidates from clusters containing more than 100 or 500 queries. The queries whose right suggestions fall in big clusters are extracted to build a small training and testing set. *Cluster* collects candidates merely based on query popularity, while *Filtering* denotes the proposed approach, which selects candidates based on both popularity and the proposed weighted similarity. We adopt the lexical features [2] and Multiple Additive Regression Trees (MART) for ranking in both methods. The results show that the proposed method outperforms the baseline obviously by kicking out unrelated queries from candidate set, and the improvement is even more significant when the cluster size is bigger.

Table 3 shows the results of the second experiment. $baseline$ only uses lexical features, $filtering$ filters unrelated candidates based on the proposed approach and uses lexical features, $filtering + length$ both filters candidates and considers query length. MART is also adopted as our ranking model, but unlike the first experiment, here we collect candidates regardless of the cluster size and NDCG@10 is used as the evaluation metric. The result shows that $filtering + length$ has improved the baseline method over 20%.

5. CONCLUSIONS

In this paper, we studied two problems of existing query suggestion methods which are based on query-URL bipartite graph. Two efficient methods, which can be easily added to existing systems, are proposed to solve the problems by filtering unrelated candidates in big clusters and taking query length as a feature. Experimental results on real dataset prove the effectiveness of the proposed models.

6. ACKNOWLEDGEMENTS

We thank the support of NSFC 91224006 and the CAS's Strategic Priority Research Program XDA06010202.

7. REFERENCES

[1] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of ACM SIGKDD*, pages 875–883. ACM, 2008.

[2] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu. Learning to suggest: a machine learning framework for ranking query suggestions. In *SIGIR'12*, pages 25–34. ACM.