A Novel Query Suggestion Method Based On Sequence Similarity and Transition Probability

Bo Shu[1], **Zhendong Niu**[1], **Xiaotian Jiang**[1], **and Ghulam Mustafa**[1]
[1]School of Computer Science, Beijing Institute of Technology, Beijing, China

Abstract— *Query suggestion plays an important role in search engines which helps to improve user experience by suggesting related terms. Conventional query suggestion methods usually employ pair-wise contextual correlation evaluation or complete sequence matching. However, when confronted with a long or newly appeared sequence, these methods cannot guarantee satisfied performance. This paper presents a novel query suggestion method to solve this problem. We first evaluate the similarities between current query sequence and each sequence in the training set, then calculate the transition probabilities from each sequence to its subsequent query. We can calculate relevance of each candidate query to the current query sequence with these interim results and retrieve the most relevant candidates for suggestion. We evaluated our method against four commonly used methods with a dataset from a commercial search engine. The experimental result shows that our method provides more relevant suggestion queries and offers better recall and precision.*

Keywords: Query suggestion, sequence similarity, transition probability

1. Introduction

Search Engines help users retrieve interesting documents. But users often input too few keywords [1] [2] and contain insufficient information for search engines to understand their intention. This is because users usually do not have a clear concept about what they want, or they use improper words to describe it. Besides, the polysemy phenomenon of words also makes it more difficult to get the exact user requirement.

Query suggestion technology solves these problems by suggesting several related query candidates to users according to their input queries, assisting them to use proper keywords to describe their search intents, and reducing the search attempts unnecessary.

There are two steps in query suggestion process: query candidates extraction, and contextual correlation evaluation. Query candidates can be extracted from both documents and query logs. When selected from documents, we often choose terms co-occurred with the current query in high ranked documents [3] [4]. This method suffers from high complexity, and the queries input by users is not employed to refine the candidates. While selected from query logs [5] [6] [7] [8], the advantages include: representing the intent of user directly, reflecting the modification process of user input queries, querying log data concisely, being easier to analysis, etc.

After selection, we need to rank the candidates in order to provide users with several of the most relevant ones, where similarity evaluation method plays an important role. There has been many researches on evaluating the correlation between query and its candidates, such as by user information [9], user feedback [10] [11] [12], arch [13], content [14], user intention [15], etc. Some methods evaluate the correlation between two queries by constructing bipartite graph with click-through data, existing user queries, and the clicked URLs [2] [6] [16] [17]. Other methods, like [18], use queries, URLs, terms to evaluate the correlation between the query and terms in document. Because the query log are very sparse, only a few popular queries link to a few high ranked URL and most URL has no query associated.

The rest of this paper is organized as follows. Section 2 discusses related works. In Section 3, we proposed our algorithm. In Section 4, we compared our method with three other existing methods, and demonstrate the experimental results. At last, we drew conclusions in Section 5.

2. Related Work

Many methods use session to cluster queries and generate query candidates [5] [19] [20] [21]. These methods need to define and identify the session [22] [23], then cluster queries with sessions and other property, such as content [24], submit time [25] [26] [27], clicked URL [28], searching topic [28], etc.

Yanan Li, et al [29] use query trace graph to calculate transition probability of queries and obtain candidates according to the probabilities. [30] improved this method with time information. However, both of the methods only calculate the transition probability of consecutive query pairs. The scarcity of taking all the prefix sequence into consideration lead to a decreasing precision in recommending queries with a long sequence.

Qi He, et al [31] proposed a new method to generate candidates. They first use the training set to generate Mixture Variable Memory Markov model, then select the candidate based on the resemblance between the user input query sequence and historical query sequence models retrieved from search engine logs.

In [31], a statistic shows that 34.34% of session patterns (specialization, generalization, parallel movement, and others) is related to the order of query session. To find out the relation between the length of sequence and the session pattern, we employed similar categories and performed similar experiments for each of the length from 2 to 5. The session set we used contains 2,000 unique sessions. Figure 1 illustrates a similar result from that in [31], which shows that 34.34% of session patterns are related to the order of query session (spelling change, generalization, and specialization). Figure 1 support the following observations: as the session length increases, roughly the proportion of specialization pattern tends to decrease, the proportion of parallel movement pattern tends to increase, the generalization pattern stays low, and the proportion of other patterns keeps fluctuating.

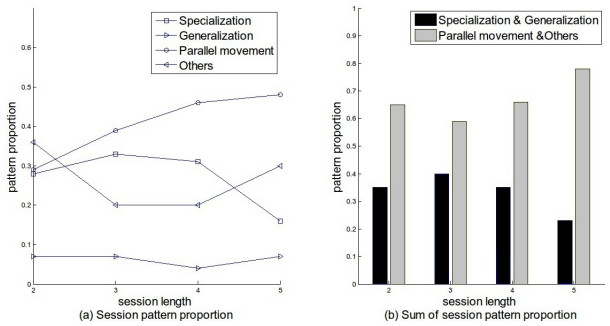

Fig. 1: Relationship between session length and session patterns

In our approach, we follow the idea of retrieving query candidates from query logs for its effectiveness and high efficiency. Some methods, such as variable memory Markov model [31], if cannot find completely matching sequence, will discard the head of sequence and try to match the rest. For example, if sequence $[q_1, q_2, \cdots, q_{i-1}]$ cannot be found in the training dataset, they remove its first element and try again to match the remaining sequence $[q_2, \cdots, q_{i-1}]$. However, according to Figure 1, relying only on complete sequence matching is not enough. Also, we need to consider the order of elements in the sequence, for the similar sequence may also provide contextual information for subsequent query suggestion. Instead of complete sequence matching along, our approach searches all the similar sequences, and measure the correlation between query sequences with similarity measures. Early researches showed that the usage of N-gram model tends to increase accuracy [31]. We also use this method and generalize it to suit the circumstance of variable length sequence - query transition probability.

We propose a novel method based on query sequence similarity and transition probability. Our method first calculate the similarity between input query and existing query sequences in the training set, then calculate the transition probability from each query to their subsequent query. After accumulating the products of each pair of similarity and transition probability to a certain subsequent query, the recommendation score of it can be obtained. According to these scores, we list the most contextual related queries to the input query as candidates.

3. Algorithm

3.1 Notation and Problem Statement

Let Q be the set of unique queries, $Q = \{q_1, \cdots, q_n\}$, S be the set of distinct sequence formed by elements in Q, $S = \{S_1, \cdots, S_m\}$, where $S_i = [q_{i1}, \cdots, q_{ij}, \cdots, q_{ik}]$, $q_{ij} \in Q$. For example, a user u may input query "computer", "DELL computer", "DELL OPTIPLEX 755", "DELL OPTIPLEX 755 price" sequentially. The query sequence of user u is S_u, $S_u = [q_1, q_2, q_3, q_4]$, q_1 is "computer", q_2 is "DELL computer", and q_3 is "DELL OPTIPLEX 755", etc.

The problem of query suggestion is to recommend candidate queries to users based on their historical query sequence. This process can be divided into 3 steps: (1) calculate the similarity between current user's input sequence and the sequence in the training set; (2) calculate the transition probability from a given query sequence to a query (both of them are in the training set); (3) for each candidate query in the training set, accumulate the products of each pair of sequence similarity and the transition probability to this query, so we can obtain the queries with the highest score as suggested queries. In the following sub-sections, we will elaborate each step in detail.

3.2 Similarity of Sequence

Many measures can be adopted to calculate the similarity of two sequences, such as Cosine Distance, Jaccard Coefficient, Hamming Distance, Minkowski Distance (including Manhattan distance and Euclid distance), Levenshtein Distance, Damerau-Levenshtein Distance, etc. Among these measures, Cosine Distance and Minkowski Distance do not concern about position information; Hamming Distance only evaluates the distance between the sequences with the same length and only adopts substitution operation; Levenshtein Distance has no transposition operation; Only Damerau-Levenshtein Distance, which is a special type of edit distance, has insertion, deletion, substitution, and adjacency transposition operation, which often been used by search engine users to modify their original input query strings. Besides, we have two other considerations: first, when two pairs of query sequences have same length, the longer common sequence the pair of query sequence has, the higher similarity it has; second, when two pairs of sequence have common sequence of the same length, the pair of sequence with the larger length has the lower similarity. Based on the above-mentioned considerations, we define the

similarity between two query sequences based on Damerau-Levenshtein Distance as:

$$sim(s_a, s_b) = 1 - (DL(s_a, s_b)/MaxLen(s_a, s_b)) \quad (1)$$

where $sim(s_a, s_b)$ is the similarity between s_a and s_b, $DL(s_a, s_b)$ is the Damerau-Levenshtein Distance between s_a and s_b, and $MaxLen(s_a, s_b)$ is the length of the longer sequence of s_a and s_b. The range of this similarity measure is $[0, 1]$ and a larger value means the two query sequences are more similar. Similar to [24], we set the threshold to 0.4, which means we regard the two sequences having a similarity less than 0.4 as completely irrelevant sequences and do not choose those sequences for the further calculation.

3.3 Transition Probability from sequence to query

By analyzing query logs, we can observe one query sequence may be followed by different queries, while different query sequences may lead to the same subsequent query. This can be depicted in a sequence-query bipartite graph with vertices representing query sequences and their subsequent queries.

The bipartite graph can be constructed with the following approach: (1) First, segment and extract sessions from query log, then convert each session into a query sequence; (2) For each sequence, we extract each possible pair of sub-sequence and subsequent query. Supposing we have the sequence $[q_1, q_2, q_3, q_4, q_5]$. It can be decomposed into the following 10 sequence-query pairs, with each of them having an occurrence weight of 1: $\langle [q_1], q_2 \rangle$, $\langle [q_1, q_2], q_3 \rangle$, $\langle [q_1, q_2, q_3], q_4 \rangle$, $\langle [q_1, q_2, q_3, q_4], q_5 \rangle$, $\langle [q_2], q_3 \rangle$, $\langle [q_2, q_3], q_4 \rangle$, $\langle [q_2, q_3, q_4], q_5 \rangle$, $\langle [q_3], q_4 \rangle$, $\langle [q_3, q_4], q_5 \rangle$, $\langle [q_4], q_5 \rangle$. (3) Search each of the pairs in the bipartite graph. If the sub-sequence or its subsequent query does not exist, we add this node in the graph. Then we attempt to add an edge to connect the sub-sequence node to the subsequent query node with the weight of 1; but if both of them are already in the bipartite graph, we simply increase the weight of their edge by 1. A brief algorithm of the bipartite graph construction process is formalized as below.

Algorithm 1: Query sequence subsequent query bipartite graph construction

Input:

T: Session training set.

Output:

G: Sequence-query transition probability bipartite graph.

Notation:

s: Session sequence in training set.

l_s: Length of sequence, i.e. the number of queries in session s.

$node\,[s]$: Node of sequence s.

$node\,[q]$: Node of query q.

$edge\,\langle s, q \rangle$: Edge from sequence s to query q.

$\phi\,\langle s, q \rangle$: Weight of the edge from sequence s to query q.

for each s in T
  for $i = 0, \ldots, (l_s - 2)$
    for $j = 2, \ldots, l_s - i$
      for $k = i, \ldots, j + i - 2$
        retrieve $[q_i, \cdots, q_(j + i - 2)]$ as query sequence
        retrieve $q_{(j+i-1)}$ as subsequent query
        if $node\left[q_i, \cdots, q_{(j+i-2)}\right]$ **not exist in** G
          add $node\left[q_i, \cdots, q_{(j+i-2)}\right]$ to G
        if $node\left[q_{(j+i-1)}\right]$ **not exist in** G
          add $node\left[q_{(j+i-1)}\right]$ to G
        if $edge\left\langle \left[q_i, \cdots, q_{(j+i-2)}\right], q_{(j+i-1)} \right\rangle$ **not exist in** G
          add $edge\left\langle \left[q_i, \cdots, q_{(j+i-2)}\right], q_{(j+i-1)} \right\rangle$ to G
          $\phi\left\langle \left[q_i, \cdots, q_{(j+i-2)}\right], q_{(j+i-1)} \right\rangle = 0$
        $\phi\left\langle \left[q_i, \cdots, q_{(j+i-2)}\right], q_{(j+i-1)} \right\rangle + +$
return G

In the bipartite graph, each sequence node is connected to multiple subsequent queries by edges. That means, for a user whose current query sequence matches a certain one in the training set, this graph provides us with his possible choices for the next query. The transition probabilities indicate these possibilities, which is evaluated by dividing the weights of the edges of each sequence node by the sum of all the weights of these edges. That is,

$$P(q|s) = \phi(s, q)/(\sum \phi(s)) \quad (2)$$

where the $P(q|s)$ is the transition probability from sequence s to its subsequent q, $\phi(s, q)$ is weight of edge from s to q, $\phi(s)$ is the weight of edge out of s. For example: the occurrence number of a sequence-query tuple $< s, q >$ is x and the occurrence number of sequence S in all sequence-query tuples is y, then in bipartite graph $\phi(s, q)$ is x and $\sum \phi(s)$ is y. The transition probability from s to q is x/y. The range of transition probability is $[0, 1]$. A larger transition probability means the user who has input a certain query sequence is more likely to choose the query connected by this edge as subsequent query.

3.4 Recommendation Score of a Query to a Sequence

The recommendation score $R\,(s, q)$ of a query to a current user query sequence is formed by accumulating the products of each pair of the similarity between the current sequence and the sequence in the training set and the transition probability from the sequence in training set to this query. Formalized by the following formula:

$$R\,(s, q) = \sum_{s_t \in T} sim\,(s, s_t) \times (sim\,(s, s_t))^{\rho - 1} \times (P\,(q|s_t)) \quad (3)$$

where the $R(s, q)$ is the recommendation score of query q to the user inputted query sequence s, s_t is a query sequence in training set T. ρ is the case amplification power [32]that modifies the influence of the similarity to the score by punishing low similarity and reducing noise. Typical $\rho \geq 1$. A large ρ makes the similarity factor less influential to the recommendation score. In this paper we set ρ to 2.5 according to [32] [33].

The diagram of sequence similarity model is shown in Figure 2. We can see that it is a directed acyclic graph and the recommendation score of a query for a input sequence is equal to the sum of path products that from input sequence to this query in training set.

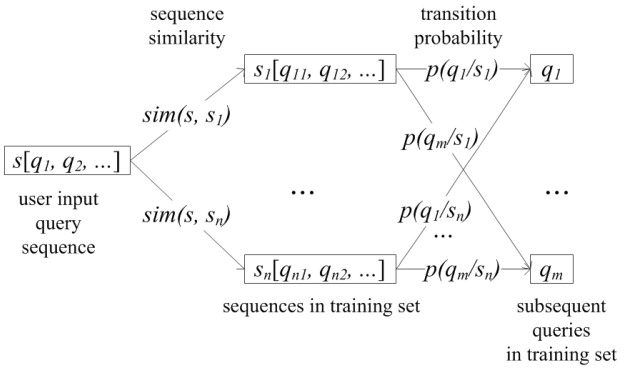

Fig. 2: Evaluation model of sequence similarity model

4. Experiment Result

We evaluated the performance of our method against four other existing query suggestion methods: text similarity [24], co-occurrence [5], query trace graph [29], and VMM[31].

4.1 Train Data Set

We employed a 30 days query log extracted from a commercial search engine (http://www.sogou.com/labs/dl/q-e.html. Corpus Search Engine Click-through Log(SogouQ). 2012-12-15.) to test our suggestion model. This is a chinese search engine and most of queries in the log are in chinese. Table 1 shows its format. Among them user ID is automatically assigned by the search engine according to the Cookie information when the users get access to it.

The input queries in the period from a user's start browsering to end has the same unique user ID. We only use user ID and query content fields in our experiment. In this 30 days query log data, we use the earlier 25 days log data as the training set and the rest as the test set.

Table 2 summarizes the statistics of the training dataset and the test dataset. The number of unique queries is the number of total query records after removing the adjacent same query records and a few unrecognizable code.

Table 1: Format of query log.

user ID	query content	...	clicked URL	...
xxx	q_1	...	www.a.com	...
yyy	q_2	...	www.b.com	...

Table 2: Statistic of query log.

Data	Searches	Unique queries	Query sequences
training	18,506,239	9,203,195	6,043,848
test	2,920,724	1,492,460	973,976

4.2 Session Segmentation and Selection

Session is defined as a sequence of queries input by a user for a specific search purpose [5]. Sessions are identified by cookie information. A cookie includes user ID, access timestamps, query text, clicked URL, etc. In practice, a session can be retrieved from search log of a search engine system, or from the query sequence by input a user with a terminal in the period from starting a browser to closing it. We identify a query sequence by user ID, and consider the query sequence as a session.

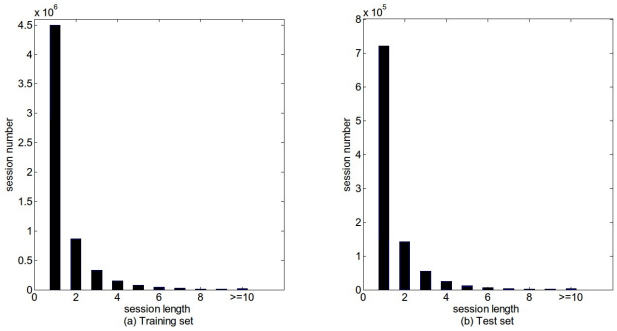

Fig. 3: Relationship between the counts and the length of the sequences in training set and test set

Figure 3 shows the relationship between the counts and the length of sequences in the training set and test set. We can see from it that the number of sequences decreases as the length of sequence goes up. 74% of sequences contain only one query. The proportions of the Sequences whose length larger than 5 are less than 1%. The sum of proportions of sequences whose length large than 5 is less than 2%, which can be safely discarded. So we selected the sequences with the length varying from 2 to 5 for training and test.

4.3 Baseline Methods

In order to evaluate the performance of the proposed method, we implement four widely-used query suggestion methods as baselines: text similarity [24], co-occurrence [5], trace graph [29], and VMM [31].

1 Text Similarity

For each query sequences we use (1) [24] to calculate the similarity between queries, where s_a and s_b denote query strings instead of query sequence. Then we select the queries that have high similarity with the user input query as the suggestion query candidates. Only the last query of the user input query sequence is considered. In the following experiment, when evaluating the recall and precision of text similarity method, we set the threshold to 0.4 [24]. That means if the similarity of a query in the training set and the user query is less than 0.4, then the former query will not be considered as a candidate.

2 Co-Occurrence

The method in [5] only evaluates the pair-wise similarity between two queries. However, in our experiment we need to evaluate the recommendation score of a query to a test sequence. So when choosing recommendation queries, we only search the queries that had co-occurred with all the queries that appears in the test query sequence in the training set. We sum up the co-occurrence between the recommendation query and each query in test query sequence as the recommendation score of recommendation query, then we select the queries with high recommendation score as candidates.

3 Trace Graph model & VMM model

We also use the trace graph model and Variable Memory Markov (VMM) Model as the baseline to evaluate our model. The details of those two model generation methods can be found in [29] and [31].

4.4 User Evaluations

We randomly selected 4000 sequences from the test set, with their lengths varying from 2 to 5. The numbers of different length sequences are approximately the same. Then we discarded the last query of each sequence to form test sequences. For each test sequence, we calculated its recommendation score with each of the five methods and chose up to 5 the most recommendable queries (Some methods, such as co-occurrence, might not generate enough candidates for a long sequence) that did not appear in the test sequence for suggestion.

20 volunteers had been chosen to evaluate the suggested queries. Each of them selected one twentieth of the total query suggestion results. After their inputting the test sequence as user query sequence, if they think the user will choose the suggested queries, then they approve it; otherwise they reject it. The volunteers not only need to consider the contextual or semantic relation between test sequence and the suggested queries, but also the intention of users after inputting the sequence. For example, the user is likely to select "DELL OPTIPLEX 755" after inputting "computer" and "DELL computer", but they seldom select "computer" after inputting "DELL computer" and "DELL OPTIPLEX 755", although they have semantic or contextual relationship.

We use the standard metrics - precision and recall - to evaluate the performance of the five query suggestion methods. Precision is defined as the ratio of the number of queries approved by volunteers over the number of all candidates. Recall is defined as the ratio of the number of queries approved by volunteers over the expected number of candidates.

4.5 Query Suggestion Recall

Figure 4 shows the recall measure achieved by the five query suggestion methods. From the results, we make the following analysis:

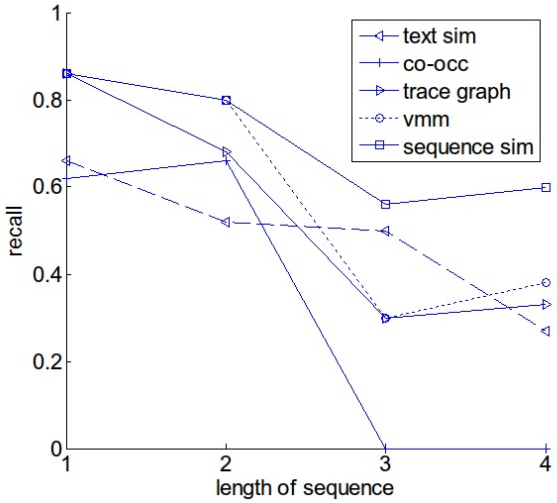

Fig. 4: Recall of five query suggestion models

1. When the length of sequence is equal to 1, the best recall is achieved by sequence similarity, trace graph, and VMM methods with the value of 0.86 because they apply the same candidate generating method.

2. As the length of sequence get larger, the recall of co-occurrence method decreases. That is ascribed to the decreasing occurrence probability of a number of queries occurred in a session as the number of queries increases. The recall of text similarity method continuously decreases, for the text similarity method only considers the last query, but a text similar query to the last query has less contextual relation with a long sequence. The trace graph, VMM, and sequence similarity methods share the same change trend. When session length is larger than two, the recall of trace graph and VMM methods are very close. That indicates VMM method tends to degenerate to trace graph method when it is more difficult to find the complete matching sequence with the increasing sequence length.

3. Sequence similarity method achieves the best recall across all sequence lengths. Because sequence similarity method will search similar sequences if it cannot find the

complete matching sequence. Thus it can always find enough candidates.

4. When the length of sequence is larger than 2, the recall of co-occurrence method drops to 0. Further analysis shows this is because co-occurrence method can find no sequence in the training set that has all the queries appeared in test sequence and still has a query that is different from the queries in test sequence for suggestion.

4.6 Query Suggestion Precision

Figure 5 shows the precision achieved by five query suggestion methods. From the results, we make the following analysis:

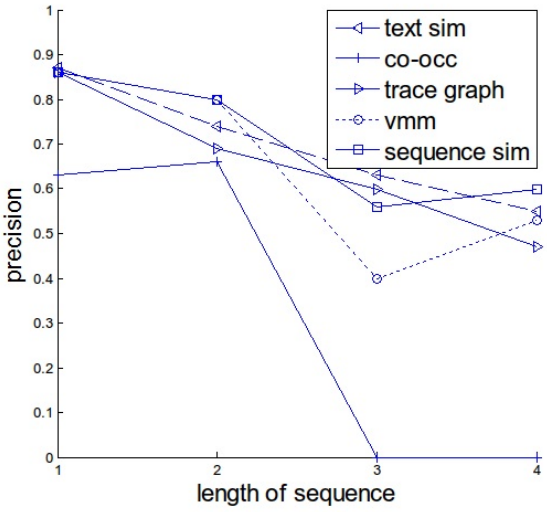

Fig. 5: Precision of five query suggestion models

1. The precision of each method has a downtrend as the length of sequence increases. Co-occurrence method has the lowest precision. When the length of sequence is larger than 2, its precision drops to 0 due to the same reason mentioned in the previous sub-section.

2. The precisions of sequence similarity, text similarity, trace graph, and VMM methods are very close when the length of sequence is equal to 1. Combining with the recall analysis, we know that is because the other three methods retrieved fewer candidates than that of the sequence similarity method. They may merely have few hits, but the less quantity of suggested queries gets their precisions raised.

3. The sequence similarity and VMM method have roughly the same change trend, because they all adopt the sequence information to generate candidates. The sequence similarity method has a higher precision due to its more flexible matching strategy.

Figure 6 shows the overall precision and recall of the user evaluation on the five query suggestion methods. We can see that although the precision of sequence similarity, text

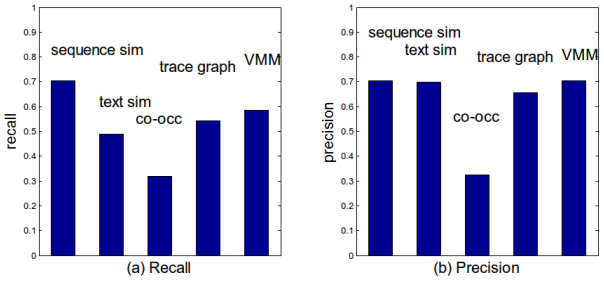

Fig. 6: Precision and Recall of volunteers' evaluation of five query suggestion models

similarity, trace graph, and VMM models are very close, the recall of sequence similarity method has an obvious advantage over the other four methods. The sequence similarity model has an outstanding comprehensive performance. We also can see from Figure 4 and Figure 5 that with the increase of the length of query sequences, the advantage of our method is more obvious.

5. Conclusions

In this paper, we present a novel method based on sequence similarity and transition probability for query suggestion. We evaluated the performances of our method with a dataset by comparing with four other existing methods. The experiment result shows that our method has both high precision and recall. This is because we adopt not only the query position information in a sequence and transition probability from query sequence to the subsequent query, but also employ a more flexible sequence matching strategy.

As future work, we will test our method with a larger training set and then extend our similarity calculation by adopting more attributes such as the time, IP, region of a user's query submission etc. Using more attributes of a query makes it more precise to evaluate the similarity between sequences.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (no. 61250010), the Program for Beijing Municipal Commission of Education (grant no.1320037010601), the 111 Project of Beijing Institute of Technology and the New Century Excellent Talents in University(grant no. NCET-06-0161).

References

[1] B. Jansen, A. Spink, and T. Saracevic, "Real life, real users, and real needs: a study and analysis of user queries on the web," *Information processing & management*, vol. 36, no. 2, pp. 207–227, 2000.

[2] J. Wen, J. Nie, and H. Zhang, "Clustering user queries of a search engine," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 162–168.

[3] Y. Qiu and H. Frei, "Concept based query expansion," in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1993, pp. 160–169.

[4] T. Cohen and D. Widdows, "Empirical distributional semantics: Methods and biomedical applications," *Journal of biomedical informatics*, vol. 42, no. 2, p. 390, 2009.

[5] C. Huang, L. Chien, and Y. Oyang, "Relevant term suggestion in interactive web search based on contextual information in query session logs," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 7, pp. 638–649, 2003.

[6] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," in *Current Trends in Database Technology-EDBT 2004 Workshops*. Springer, 2005, pp. 395–397.

[7] C. Huang, L. Chien, and Y. Oyang, "Clustering similar query sessions toward interactive web search."

[8] S. Jiang, S. Zilles, and R. Holte, "Query suggestion by query search: a new approach to user support in web search," in *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, vol. 1. IET, 2009, pp. 679–684.

[9] P. Chirita, C. Firan, and W. Nejdl, "Personalized query expansion for the web," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 7–14.

[10] L. Fitzpatrick and M. Dent, "Automatic feedback using past queries: social searching?" in *ACM SIGIR Forum*, vol. 31, no. SI. ACM, 1997, pp. 306–313.

[11] M. Magennis and C. van Rijsbergen, "The potential and actual effectiveness of interactive query expansion," in *ACM SIGIR Forum*, vol. 31, no. SI. ACM, 1997, pp. 324–332.

[12] Y. Song and L. He, "Optimal rare query suggestion with implicit user feedback," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 901–910.

[13] R. Kraft and J. Zien, "Mining anchor text for query refinement," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 666–674.

[14] J. Wen, J. Nie, and H. Zhang, "Query clustering using user logs," *ACM Transactions on Information Systems*, vol. 20, no. 1, pp. 59–81, 2002.

[15] M. Strohmaier, M. Kröll, and C. Körner, "Intentional query suggestion: making user goals more explicit during search," in *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 2009, pp. 68–74.

[16] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 407–416.

[17] 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 *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 875–883.

[18] M. Diligenti, M. Gori, and M. Maggini, "Users, queries and documents: A unified representation for web mining," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 2009, pp. 238–244.

[19] B. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani, "Concept-based interactive query expansion," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 696–703.

[20] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 387–396.

[21] B. Fonseca, P. Golgher, E. De Moura, B. Pôssas, and N. Ziviani, "Discovering search engine related queries using association rules," *Journal of Web Engineering*, vol. 2, no. 4, pp. 215–227, 2003.

[22] D. He, A. Göker, and D. Harper, "Combining evidence for automatic web session identification," *Information Processing & Management*, vol. 38, no. 5, pp. 727–742, 2002.

[23] A. Jansen, B. J. Spink, C. Blakely, and S. Koshman., "Defining a session on web search engines," *Journal of The American Society for Information Science and Technology*, vol. 58, no. 6, pp. 862–871, 2007.

[24] X. Shi and C. Yang, "Mining related queries from search engine query logs," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 943–944.

[25] S. Chien and N. Immorlica, "Semantic similarity between search engine queries using temporal correlation," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 2–11.

[26] Q. Mei, D. Zhou, and K. Church, "Query suggestion using hitting time," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 469–478.

[27] R. Baraglia, C. Castillo, D. Donato, F. Nardini, R. Perego, and F. Silvestri, "Aging effects on query flow graphs for query suggestion," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1947–1950.

[28] D. Widyantoro and J. Yen, "Using fuzzy ontology for query refinement in a personalized abstract search engine," in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, vol. 1. IEEE, 2001, pp. 610–615.

[29] Y. Li, B. Wang, S. Xu, P. Li, and J. Li, "Querytrans: Finding similar queries based on query trace graph," in *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, vol. 1. IET, 2009, pp. 260–263.

[30] R. Baraglia, F. Nardini, C. Castillo, R. Perego, D. Donato, and F. Silvestri, "The effects of time on query flow graph-based models for query suggestion," in *Adaptivity, Personalization and Fusion of Heterogeneous Information*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2010, pp. 182–189.

[31] Q. He, D. Jiang, Z. Liao, S. Hoi, K. Chang, E. Lim, and H. Li, "Web query recommendation via sequential query prediction," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 2009, pp. 1443–1454.

[32] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[33] D. Lemire, "Scale and translation invariant collaborative filtering systems," *Information Retrieval*, vol. 8, no. 1, pp. 129–150, 2005.