ABSTRACT
In this work we propose a method that retrieves a list of related queries given an initial input query. The related queries are based on the query log of previously issued queries by human users, which can be discovered using our improved association rule mining model. Users can use the suggested related queries to tune or redirect the search process. Our method not only discovers the related queries, but also ranks them according to the degree of their relatedness. Unlike many other rival techniques, it exploits only limited query log information and performs relatively better on queries in all frequency divisions.

Categories and Subject Descriptors
H.3.5 [Information Storage and Retrieval]: Online Information Services – Web-based services

General Terms: Algorithms, Experimentation

Keywords
Association rule, related query, edit distance, query log, web searching

1. INTRODUCTION
Web search engines have become the most popular solution to finding relevant information to a topic on the web. However, search engine users often experience difficulties in organizing and representing their information needs by simple queries. It is often desirable for search engines to give suggestions on similar and related queries to users’ input queries. Besides, discovered related queries can also be further used for query expansion or searching optimization. Some recent works [1, 2] have made good attempts in mining related queries from the search engine query logs and some of the results were promising. In this work, we propose to use an improved association rule mining model to mine related queries from query transactions in query logs. We also propose a simple but effective segmentation algorithm that segments user sessions into query transactions.

2. DEFINITIONS
We present the definitions of key terminologies in this section.

Query Record: A query record represents the submission of one single query from a user to the search engine at a certain time. It can be represented as a set of triplets \( I_i = (q_i, ip_i, t_i) \), where \( q_i \) is the submitted query (i.e. terms), \( ip_i \) is the IP address of the host from which the user issues the query, and \( t_i \) represents the timestamp when the user submits that query.

Query Transaction: A query transaction is the search process 1) with the search interest focusing on the same topic or strongly related topics, 2) in a bounded and consecutive period, and 3) issued by the same user. It is represented as a series of query records in temporal order, i.e. \( T_j = \{I_1, I_2, \ldots, I_m\} = \{(q_1, ip_1, t_1), (q_2, ip_2, t_2), \ldots, (q_m, ip_m, t_m)\} \) where \( ip_1 = ip_2 = \ldots = ip_m \) and \( t_1 \leq t_2 \leq \ldots \leq t_m \).

User Session: A user session contains the history of all query records that belong to the same user, in the query log. It can be represented as a series of query records in temporal order, i.e. \( S_i = \{I_1, I_2, \ldots, I_n\} = \{(q_1, ip_1, t_1), (q_2, ip_2, t_2), \ldots, (q_n, ip_n, t_n)\} \) where \( ip_1 = ip_2 = \ldots = ip_n \) and \( t_1 \leq t_2 \leq \ldots \leq t_n \) and \( n \geq m \).

Given these definitions, we have the following constraints:
\[
\forall i \exists j, k \ I_i = I_j = I_k \leq S_i
\]
(1)
\[
\forall i, j, k \ I_i \neq \emptyset, \ S_i \neq \emptyset
\]
(2)
\[
\forall i, j, p, q \ I_i \cap I_j = \emptyset, \ S_i \cap S_j = \emptyset
\]
(3)
\[
\forall j \exists k \ T_j \subseteq S_k
\]
(4)

3. LEVENSHTEIN DISTANCE SIMILARITY
Because search engine users often reformulate their input queries by adding, deleting or changing some words of the original query string, we use Levenshtein distance [3], which is a special type of edit distance, to measure the degree of matching between query strings. It defines a set of edit operations, such as insertion or deletion of a word, together with a cost for each operation. The distance between two query strings then is defined to be the sum of the costs in the cheapest chain of edit operations transforming one query string into the other. For example, the Levenshtein distance between “adobe photoshop” and “photoshop” is 1.

Hence the similarity between two queries can be measured by the Levenshtein distance similarity between them and defined as:
\[
similarity_{Levenshtein}(q_1, q_2) = 1 - \frac{\text{Levenshtein distance}(q_1, q_2)}{\max(wn(q_1), wn(q_2))}
\]
(5)
where \( \text{wn}() \) is the number of words (or characters for Chinese queries) in a query.

The Levenshtein distance similarity is seldom applied to finding related queries because it retrieves only highly matching queries and thus fails to discover those related queries that are dissimilar in their terms, e.g. “search engine” and “google”.

4. SEGMENTATION ALGORITHM
Our proposed model is based on the traditional association rule mining technique. For mining association rules of queries, we need to statistically measure the co-occurrences between queries in query transactions; so the quality of segmenting user sessions into query transactions is critical for mining related queries.

We developed a dynamic sliding window segmentation algorithm that adopts three time interval constraints, i.e. 1) the maximum interval length allowed between adjacent query records in a same query transaction (\( \alpha \)), 2) the maximum interval length of the period during which the user is allowed to be inactive (\( \beta \)), and 3) the maximum length of the time window the query transaction is allowed to span (\( \theta \) (\( \alpha \leq \gamma \leq \beta \)). It also sets a lower bound for the Levenshtein distance similarity between adjacent queries, i.e. \( \theta \), to justify the borders of query transactions. We empirically set the
values of α, β, γ, θ to be 5 minutes, 24 hours, 60 minutes and 0.4 in our experiments. The complexity of this algorithm is O(n). Figure 1 shows the pseudo-codes for this segmentation algorithm.

**Input:** A set of user sessions $S = \{S_1, S_2, S_3, \ldots, S_n\}$ where $S_i$ is a series of query records in temporal order, i.e., $\{I_{l_1}, I_{l_2}, \ldots, I_{l_m}\} = (q_{t_1}, q_{t_2}, I_{l_1}), (q_{t_2}, q_{t_3}, I_{l_2}), \ldots, (q_{t_m}, q_{t_{m+1}}, I_{l_m})$

**Output:** A set of query transactions $T = \{T_1, T_2, T_3, \ldots, T_s\}$.

**Procedure SEGMENT transaction set $T' \rightarrow \Phi$**

- sort $S$ in temporally ascending order
- for each $S_i$ in $S$
  - transaction $t \leftarrow new\_empty\_transaction$
  - append to $T$
  - timestamp of previous query record $t_{prev} \leftarrow t_i$
  - start time of current transaction $t_{cur\_trans} \leftarrow t_i$
  - for each $I_l$ in $S_i$
    - if $t_l - t_{prev} \leq \alpha$ and $t_l - t_{cur\_trans} \leq \gamma$ then
      - append $I_l$ to $t$
    - else if $t_l - t_{prev} > \beta$ then
      - append to $T$
    - else
      - find the last query record $l_{last} = \tau$, i.e., closest to $t_l$
      - $t \leftarrow new\_empty\_transaction$
      - append $t$ to $T$
      - $t_{cur\_trans} \leftarrow t$
      - append $I_l$ to $t$
  - return $T$

Figure 1. Dynamic Sliding Window Segmentation Algorithm

5. MINING RELATED QUERIES

Our model is a modified-confidence version of the traditional approach of mining association rules. Here we define $Q = \{q_1, q_2, q_3, \ldots, q_n\}$ as the set of unique queries from query log files and $T$ is the set of query transactions $t$. For each $t$ there is a binary vector $I[k]$ such that $I[k] = 1$ if query transaction $t$ contains query record $q_k$ that searched for query $q_k$, and $I[k] = 0$ otherwise. Let $X$ be a non-empty subset of $Q$. A transaction $t$ satisfies $X$ if for all queries $q_k$ in $X$, $I[k] = 1$.

The association rule is redefined to mean an implication $X \Rightarrow q_j$, where $X \subseteq Q$, and $q_j \notin X$. As we are interested only in finding related queries given an initial input query, the set $X$ contains only the initial input query $q_j$, i.e., $X = \{q_j\}$. Therefore the association rule in this problem becomes $q_j \Rightarrow q_j$, where $q_j \in Q$. $q_i \in Q$ and $i \neq j$. Mining related queries is simplified as finding the statistical associations between the input query and any other queries, hence.

The association rule $q_j \Rightarrow q_j$ has a support factor of $s$ if $s\%$ of the transactions in $T$ satisfy both $\{q_j\}$ and $\{q_j\}$, noted as $q_j \Rightarrow q_j | s$. We define the raw confidence factor of the association rule $q_i \Rightarrow q_j$ to be $rc$ if $rc\%$ of the transactions in $T$ satisfy $\{q_j\}$, given that $T'$ is the set of all transactions in $T$ that satisfy $\{q_j\}$, and is noted as $q_j \Rightarrow q_j | rc$. Then we combine the raw confidence factor with the Levenshtein distance similarity between $q_i$ and $q_j$.

The final confidence factor of $q_j \Rightarrow q_j$ is calculated as:

$$ (q_j \Rightarrow q_j | c) = \frac{q_j \Rightarrow q_j | rc \times e^{similarity(\text{Levenshtein}(q_j, q_j))}} {\text{support}(q_j)} $$

(6)

Assuming the input query is $q_j$, we calculate the support factor $q_j \Rightarrow q_j | s$ and confidence factor $q_j \Rightarrow q_j | c$ of any hypothesized association rule $q_i \Rightarrow q_j$. Then we first set a threshold minsupport for the support factors to filter away those association rules that are not statistically strong enough. Next we rank the list of association rules according to their confidence factors. Finally we select the top $K$ queries (if available) in the list and return them as the most related queries to the input query $q_j$.

The Levenshtein distance similarity is introduced as a non-penalizing decaying factor in (6), which is non-linear. We found that the traditional association rule mining model favors frequent queries and often fail to retrieve infrequent queries that are highly similar to the input query. The non-linear non-penalizing decaying factor promotes the positions of those queries in the ranked list without penalizing others significantly.

6. EXPERIMENTS

We have tested our method on a dataset collected from the query logs of Tianwang (www.tianwang.com) search engine. It covers 4 months from March 2003 to June 2003 and about 80% of the queries in it contain Chinese words. Approximately 14 million query records and 3 million distinct queries are identified.

We selected 100 test input queries “randomly” according to the overall frequency distributions. The frequencies of these test input queries range from 50 to 75,975 evenly. We selected the top 20 queries, if available, for experimental evaluations. Overall precision rates were calculated after the relatedness of retrieved queries was evaluated by a group of three human annotators.

We compare our improved association rule mining model with three rival models including 1) temporal correlation model (TCM) as the baseline, 2) association rule mining model [1] (ARM) and 3) our improved association rule mining model (ARM_LDS). We also compare our dynamic sliding window segmentation algorithm (DSW SA) with the naïve segmentation algorithm (Naïve SA) proposed in Fonseca, et al. [1]. The experimental results are presented in Table 1 below.

<table>
<thead>
<tr>
<th>Top K Queries</th>
<th>TCM</th>
<th>ARM</th>
<th>ARM_LDS</th>
<th>ARM</th>
<th>ARM_LDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56.65</td>
<td>91.86</td>
<td>94.65</td>
<td>95.35</td>
<td>97.65</td>
</tr>
<tr>
<td>5</td>
<td>60.47</td>
<td>85.60</td>
<td>89.73</td>
<td>90.88</td>
<td>93.64</td>
</tr>
<tr>
<td>10</td>
<td>54.98</td>
<td>81.11</td>
<td>85.44</td>
<td>88.45</td>
<td>90.59</td>
</tr>
<tr>
<td>15</td>
<td>50.63</td>
<td>75.76</td>
<td>80.88</td>
<td>86.05</td>
<td>89.88</td>
</tr>
<tr>
<td>20</td>
<td>44.32</td>
<td>71.66</td>
<td>76.29</td>
<td>83.29</td>
<td>88.44</td>
</tr>
</tbody>
</table>

7. CONCLUSIONS

In this paper we propose a method of automatically discovering related queries from web search engine query logs. This method first segments the user sessions identified in query logs into query transactions, and then mines association rules of related queries using an improved association rule mining model which utilizes not only the co-occurrences between distinct queries but also the Levenshtein distance similarity between them. The experimental result shows that our method obtained approximate gains (in precision rates with $K = 20$) 17% and 44% compared with rival models and the baseline respectively.

REFERENCES

