A Personalized Query Suggestion Agent based on Query-Concept Bipartite Graphs and Concept Relation Trees

Yan Chen* and Yan-Qing Zhang

Department of Computer Science, Georgia State University, Atlanta, GA 30302-3394, USA.
E-mail: ychen44@student.gsu.edu
E-mail: yzhang@cs.gsu.edu
*Corresponding author

Abstract: Queries submitted to a Web search engine are usually short and ambiguous. Currently, most search engines respond to a user’s query by using the bag-of-words model, which matches keywords between the query and Web documents but ignores contexts and users’ preferences. Thus, many irrelevant results are returned by the conventional search engines. Query suggestion is a way for extending queries to allow search engines to better speculate exact meanings of short and ambiguous queries. This paper proposes a personalized query suggestion agent that uses the Query-Concept bipartite graphs and Concept Relation Trees for query suggestion. Firstly, the personalized query suggestion agent uses both concepts’ semantic relations and concepts’ co-occurrence for concept clustering to have better performance. Secondly, the agent constructs Concept Relation Trees that can provide more suggested queries than a Query-Concept based method. Thirdly, the agent dynamically updates weights between Query-Concept and Concept-Concept to personalize suggestions. Simulation results show that the new personalized query suggestion agent works effectively.

Keywords: concept relation trees; bipartite graph; semantic relations; co-occurrence; personalization.

Reference to this paper should be made as follows:

Biographical notes: Yan Chen is a Ph.D student of Computer Science at the Georgia State University, United States. His research interests include semantic computing, data mining and information retrieval.

Yan-Qing Zhang received the Ph.D degree in Computer Science from the University of South Florida, United States. He is an Associate Professor of Computer Science at the Georgia State University. His research interests include computational web intelligence, data mining and knowledge discovery.

1 Introduction

With the rapid growth of information on the Web, the search engines that help users exploit such an extremely valuable resource have gained great momentum both in the
Yan Chen and Yan-Qing Zhang

academic and commercial areas. Currently, most search engines respond to a user’s query by using the Bag-of-words model (Yang et al, 2007), which matches keywords between the query and Web documents. Queries usually contain ambiguous terms, whose exact meanings are determined by the contexts. However, the Bag-of-words model simply treats text as an unordered collection of words, so the contexts of queries are ignored. For instance, Bag-of-words model cannot tell the difference for “apple” between the query “apple pie” and the query “apple computer.” A long query may contain enough contexts for helping the search engines determine the exact meaning for ambiguous queries. However, based on the research of Kules, Kustanowitz, and Shneiderman (2006), the average length of queries submitted to World Wide Web search engines is only two words, which makes it difficult to speculate the meaning of the queries.

The first frequently used technique for dealing with ambiguous and short queries is the Query Suggestion, such as Yahoo Search Assist, which extracts terms semantically related to queries, and then suggests users to reformulate their queries based on those terms. This technique extends the short and ambiguous queries by adding semantically related contexts. Therefore, the machines can speculate the meanings of ambiguous queries easier. However, these current search engines simply suggest some semantically related terms to the queries, but do not consider users’ preferences. Thus, different users get the same suggestions although they may have different interests.

The second frequently used technique is Web Search Results Classification, which categorizes the search results based on a predefined concept hierarchy (Ricardo, Carlos, and Marcelo, 2004). A user can easily select one cluster of Web pages based on what he or she needs. However, because the hierarchy is predefined, it may be too coarse and may not contain a category that best represents user’s personal interests. For example, if a user inputs a query “car”, he or she may be interested in “car rental” or “car accident” category, but not in “car vehicle” category.

The third frequently used technique is the Personalized Query extension, which extends the current query by appending relevant past queries (Wang and Zhai, 2007). For example, based on users’ past visiting history, we found the query “jaguar” and another query “car fix” are relevant. Then, if that user inputs “jaguar”, then the extended query “jaguar car fix” will be created as a personalized query suggestion. Because the extended query is highly related to the past queries, it should be much more relevant to the users’ interests.

This paper presents a user-oriented, concept-based query suggestion agent. Firstly, the clickthrough data sets that contain users’ queries and corresponding clicked URLs will be analyzed by our model, and concepts related to the queries will be extracted. Then, a Query-Concept bipartite graph will be constructed. Secondly, based on concept semantic relations and co-occurrence frequencies, the extracted concepts are clustered and thus Concept Relation Trees (CRT) can be constructed. The CRTs are tree-structure concept clusters in which concepts are represented as leaves and any two concepts’ relation is demonstrated as the weight of their lowest common ancestor. If queries are connected by the CRTs, their relations can be calculated based on the relations of Query-Concept and Concept-Concept obtained in the first two steps. Thirdly, we propose strategies for calculating the relations of Query-Query. Therefore, for a given query, we are able to retrieve all related queries, which have a strong relationship with that query, as suggestions. Fourthly, based on the users’ recent queries and clicked URLs, we dynamically update the weights of Query-Concept and concepts’ relations in the CRTs. The architecture of our query suggestion agent is shown in the figure 1.
2. Queries Suggestions

In the section 2.1, we introduce a well-known Query-URLs based suggestion agent first. Then, from the section 2.2, our CRT-based query suggestion agent will be presented.

2.1 Query-URLs based Query Suggestions

Based on the user’s clickthrough data, if a query results in a Web page clicking, the Query-URLs based suggestion agent (Beeferman and Berger, 2006) considers they have relations and links the query to that URL. If two queries are linked by the same URL, they will be considered relevant. Thus, for a given query linked by one URL, Query-URLs based suggestion agent simply retrieves all queries linked by the same URL as suggestions.

One major problem with the Query-URLs based suggestion agent is that the number of common clicks on URLs for different queries is limited. In a large clickthrough dataset
from a commercial search engine, it was reported that the chance for two random queries to have a common click is less than $10^{-4}$ (Beeferman and Berger, 2006). Therefore, it is very difficult to find the relevant queries for a given query.

Another problem with the Query-URLs based method is that although two queries may lead to the same URLs clicking, they may still be irrelevant because they may point to totally different contents of the Web document. For example, in the figure 2, two queries “apple pie history” and “strawberry season” lead to the same Web page “Food day” clicking, but we cannot say that those queries are related because they point to different contents of “Food day.”

2.2 Extracting Concepts from Clicked Web Pages

Based on the user’s clickthrough data, our query suggestion agent firstly constructs a Query-Document bipartite graph that links all clicked Web pages for given queries. Secondly, the agent extracts the concepts from those Web pages based on frequencies of those concepts occurring in the clicked documents. Thus, we are able to construct a Query-Web Pages-Concept tripartite graph as shown in figure 3.
A Personalized Query Suggestion Agent

For a given query, a user needs to browse several Web pages returned by the traditional search engines and locate the information that he/she needs. We assume that if the visiting period of one Web page is long enough, that page may contain the information related to the query. Therefore, the following equation $RQD(q, d_i)$ is proposed for measuring the relation of one Web page $d_i$ with the query $q$.

$$RQD(q, d_i) = \frac{\sum_{j=1}^{m} \text{period}_j(d_i)}{\sum_{j=1}^{n} \text{period}_j(d_i)}$$

In the above equation, $m$ means the user clicks the Web page $d_i$ $m$ times, and the period of $j^{th}$ visiting is $\text{period}_j(d_i)$; $n$ means the user clicks $n$ Web pages for that query in total, and the divisor of the equation is the total visiting period of the clicked Web pages for that query.

After the relations between queries and Web pages are calculated, we can extract relevant concepts for the queries based on the assumption that if a keyword or phrase appears frequently in the web pages that users select, it should be an important concept related to the query. Therefore, we propose another equation $RQC(q, c_j)$ for measuring the relation of a particular concept $c_j$ with the query $q$.

$$RQC(q, c_j) = \sum_{k=1}^{m} RQD(q, d_k) \times f_{d_k}(c_j)$$

In the above equation, $RQD(q, d_k)$ is the relation of the query $q$ and document $d_k$ that contains concept $c_j$; $f_{d_k}(c_j)$ is the occurrence frequency of the concept $c_j$ in the document $d_k$; $m$ indicates that $m$ clicked Web pages contain the concept $c_j$.

After obtaining the relations between concepts and queries, we can extract the concepts that may represent the users’ interested information for the queries if the concepts’ relations with the query are larger than the threshold $\delta_1$. For example, for the query “java”, based on the user’s clickthrough data, we may extract the related concepts from Web pages, like “programming language”, “software”, “code”, “compiler”, “technology”, “virtual machine”, “island”, “Indonesia”, “Jakarta”, “resort”, “coffee”, “boca”, “tea”, and “gourmet.” The initial value of $\delta_1$ is set as 0.01, and it can be adjusted later.

2.3 Calculating the Concepts’ Relations

After the concepts are extracted from the Web pages, those concepts may be divided into different concept sets, and each concept set represents a cluster of closely-related concepts. To cluster extracted concepts, we propose the following equation $RCC(c_i, c_j)$ to calculate the relationship between concept $c_i$ and concept $c_j$.

$$RCC(c_i, c_j) = \frac{1}{2} \times (SR(c_i, c_j) + CO(c_i, c_j))$$

Based on the above equation, two concepts’ relation $RCC(c_i, c_j)$ is determined by the concepts’ semantic relation $SR(c_i, c_j)$ and the co-occurrence frequency $CO(c_i, c_j)$. Thus, in the following parts, we present our methods for calculating the concepts’ semantic relations and co-occurrence frequencies.
WordNet [9] is a large lexical database in English, developed under the direction of George A. Miller. Synonymous words are grouped together into synonym sets, called synsets. Each synset represents a single distinct sense or concept. Each WordNet sense is associated with a tree structure in the WordNet Is-A hierarchy. The nodes in these tree structures are WordNet hyponyms, each of which has a unique identifier in WordNet. Therefore, each sense can be related to the unique hyponyms above the sense in the tree structure. In the Is-A hierarchy tree, each child node is an instance of the parent node, like “car” is kind of “vehicle”, and “vehicle” is kind of “physical entity.” A part of WordNet Is-A Hierarchy is shown in the figure 4.

The semantic similarity between concepts can be estimated by the information content (IC) \([\cdot]\). The information content of a concept \(x\) is defined as

\[
IC(x) = -\log(p(x)),
\]

where \(p(x)\) denotes the frequency of encountering an instance of concept \(x\). The frequency of encountering a concept includes the frequency of encountering all its subordinate concepts since the count we add to a concept is added to its subsuming concept as well. If the \(p(x)\) of the root node of the WordNet Is-A tree is defined as 1, for any concept node \(c\) in that tree, its \(p(x)\) can be calculated by the equation: \(n_c/n_a\), where \(n_c\) represents the number of descendants of that concept node \(c\), and \(n_a\) represents the number of all nodes in the tree. Therefore, the information content of a concept is \(-\log(n_c/n_a)\). Then, by applying the Jaccard similarity coefficient (Tan, Steinbach, and Kumar, 2005), we propose the following equation \(SR(c_i, c_j)\) to calculate any two concepts’ relation,

\[
SR(c_i, c_j) = \frac{|-\log \frac{n_{c_i}}{n_p}|}{\left|(-\log \frac{n_{c_i}}{n_p}) + (-\log \frac{n_{c_j}}{n_p}) - (-\log \frac{n_{c_j}}{n_p})\right|}.
\]

In the above equation, \(n_p\) is the number of descendants of the lowest common ancestors of \(c_i\) and \(c_j\); \(n_{c_i}\) is the number of descendants of the concept \(c_i\); \(n_{c_j}\) is the number of descendants of concept \(c_j\); \(n_a\) represents the number of all nodes in the tree.

Based on above equation, we may estimate that, in figure 4, concepts “car” and “ship” have higher semantic relation than the concepts “car” and “cabin” because the lowest common ancestor of “car” and “ship”, “vehicle”, contains fewer concepts than the lowest common ancestor of “car” and “cabin”, “physical entity”, although “car” and “cabin” have the same number of children nodes as the concept “car” and “ship.”
We propose the following equation $CO(c_i, c_j)$ to calculate the frequency of co-occurrences of concepts $c_i$ and $c_j$.

$$CO(c_i, c_j) = \frac{2 \times f(c_i \cap c_j)}{f(c_i) + f(c_j)}$$

In the equation $CO(c_i, c_j)$, $f(c_i \cap c_j)$ is the frequency of the Web pages that contain both concepts $c_i$ and $c_j$, and $f(c_i)$ and $f(c_j)$ is the frequency of the Web pages that contain the concept $c_i$.

### 2.4 Constructing the Concept Relation Trees

After concepts are extracted based on the frequencies of occurring in the users’ selected Web pages, we may group the concepts that have high co-occurrence frequencies and similar semantic relations together. Thus, based on the concepts’ relation equation $RCC(c_i, c_j)$ presented in the section 2.3, we propose an agglomerative clustering algorithm to construct concept clusters, each of which contains closely-related concepts. In the section 2.2, we have already obtained the relative concepts for the queries. Then, the queries connected by the concepts in the same cluster should have very strong relationships, so one query of them may be considered as the suggestion of another.

Before presenting the algorithm 1, we introduce the term “pseudo-concept” used in it. “Pseudo-concept” is a CRT’s node grouped by the concepts and represents the union of set concepts.

### ALGORITHM 1 Constructing CRTs

**INPUT:**
The extracted concepts, concepts’ co-occurrence frequencies and semantic relations

**OUTPUT:**
Several CRTs, each of which is a concept cluster and contains closely-related concepts

**BEGIN**

**Step 1:** Based on the equation $RCC(c_i, c_j)$, calculate the relations for all possible pairs of extracted concepts. Thus, the matrix of the concepts’ relations $M$ is created.

**Step 2:** Merge $a$ and $b$, which are disjoint concepts, pseudo-concepts or one concept and another pseudo-concept, and they have the highest concepts’ relation. Then, create a pseudo-concept $t$ to represent the union of set $a$ and $b$.

**Step 3:** Calculate the relations between that pseudo-concept $t$ with other disjoint concepts and pseudo-concepts. The relation $r$ between $t$ and another concept or pseudo-concept $t'$ is the highest concepts’ relation between the concepts from $t$ and $t'$. Then, assign $r$ to the relations between any concepts from $t$ and $t'$. Next, update the corresponding concepts’ relations in the matrix $M$.

**Step 4:** Repeat the step 2 and 3 until all the relations between any concepts or pseudo-concepts are smaller than a threshold value $\delta_2$, or all concepts are grouped into one pseudo-concept.

**END ALGORITHM 1.**

In the algorithm 1, concepts are grouped together if their relation is greater than a threshold value $\delta_2$. The initial value of $\delta_2$ in the algorithm 1 is set as 0.3, and it can be adjusted later. Based on the algorithm 1, the extracted concepts are clustered and thus CRTs can be constructed. The CRTs are tree-structure concept clusters in which concepts
Yan Chen and Yan-Qing Zhang

are represented as leaves and any two concepts’ relation is demonstrated as the weight of their lowest common ancestor. For the previous “Java” example, after the concepts related to the query “java” are extracted and their relations are calculated, based on the algorithm 1, we can create following three CRTs as shown in figure 5. The square nodes represent the concepts and the round nodes represent the pseudo-concepts. The nodes’ weights indicate the relations between concepts. For any two concepts nodes in a CRT, their relation is demonstrated as their lowest common ancestor’s weight. For example, the relation of “island” and “resort” is their lowest common ancestor’s weight 0.32. Once the CRTs are constructed, we will save them into users’ profiles, which can be used as knowledge bases for personalizing query suggestions.

Figure 5 Concept relation trees for “java”

For the concepts related to a single query, we can create multiple CRTs, each of which contains closely-related concepts. Given concepts related to two queries \( q_1 \) and \( q_2 \), we may construct separate CRTs related to them if their concepts are completely unrelated. However, if the CRTs for different queries contain the same concepts, we may need to merge the CRTs together to indicate their relations. Thus, we propose the following approaches to solve this problem in different situations.

If the concepts related to the query \( q_1 \) are a sub set of concepts related to query \( q_2 \), we only need to construct CRTs for \( q_2 \). But we still need to calculate the concepts’ relations for the \( q_1 \) based on the equation \( RCC(c_i, c_j) \), and use these relations to update concepts’ relations of the CRTs for \( q_2 \). One updated relation of the CRTs for \( q_2 \) is the average value of the old relation for \( q_2 \) and corresponding concepts’ relation for the \( q_1 \). For the above “java” example, we construct three CRTs, one of which contains the concept “tea”, “boca”, “coffee”, and “gourmet.” Then, we have another query “Starbucks”, and the concepts extracted from the users’ clicking Web pages are “coffee” and “boca.” In that situation, we do not need to create separate CRTs for the “Starbucks”, but just calculate the relation for “coffee” and “boca” based on the equation \( RCC(c_i, c_j) \), and use that relation to update the “coffee” and “boca” relation in the CRT of “java.” After the concepts’ relations updated, the structure of the CRTs need to be updated for keeping concepts’ relations in order. In other words, the concepts that have lower common ancestors should have higher relations than the concepts that have higher
A Personalized Query Suggestion Agent

common ancestors. Therefore, we propose the algorithm 2 to update CRT’s structures based on altered concepts’ relations.

ALGORITHM 2 Updating CRT based on altered concepts’ relations

INPUT:
The old CRT with altered concepts’ relations

OUTPUT:
An updated CRT with new concepts’ relations and updated structure

BEGIN

Step 1: Store the concept pairs with altered relations in an array $T$. Then, rank the altered concepts’ relations decreasingly.

Step 2: Select one concept pair ($c_i$, $c_j$) that has the highest altered relation in the $T$. Compare the altered relation of concepts ($c_i$, $c_j$) with their old relations. If the altered relation is higher than their old relation, go to step 3; otherwise, remove ($c_i$, $c_j$) with their concepts’ relation in the $T$. If no more concept pairs in $T$, then end the algorithm; otherwise, go back to step 2.

Step 3: Compare the altered relation of concepts ($c_i$, $c_j$) with the weights of $c_i$’s parent $p^1(c_i)$ and $c_j$’s parent $p^1(c_j)$. (The weight of $p^1(c_j)$ is the concepts’ relation between $c_i$ with its nearest neighbour concept.) If the altered relation of concepts ($c_i$, $c_j$) is smaller than the weights of $p^1(c_i)$ and $p^1(c_j)$, then compare the altered relation with the weights of $p^1(p^1(c_i))$ and $p^1(p^1(c_j))$, or $p^2(c_i)$ and $p^2(c_j)$. Repeat this step until the altered relation of concepts ($c_i$, $c_j$) is larger than the weight of $p^1(c_i)$ or $p^1(c_j)$, or both.

Step 4: If the altered relation of concepts ($c_i$, $c_j$) is larger than the weight of $p^1(c_j)$ but smaller than the weight of $p^1(c_i)$, break the connection between the children nodes of $p^1(c_j)$. Then, merge the $p^1(c_j)$ with the child nodes of $p^{1-1}(c_j)$. The weight for the new created connection is the altered relation of concepts ($c_i$, $c_j$). If the altered relation of concepts ($c_i$, $c_j$) is larger than the weights of $p^1(c_i)$ and $p^1(c_j)$, break the connection between the children nodes of $p^1(c_i)$ and $p^1(c_j)$. Then, merge the $p^1(c_i)$ and the $p^1(c_j)$. The weight for the new created connection is the altered relation of concepts ($c_i$, $c_j$).

Step 5: Merge the nodes whose connection are broken in step 4 with the new created nodes. Then, remove ($c_i$, $c_j$) with their concepts’ relation in the array $T$. If no more concept pairs in $T$, then end the algorithm; otherwise, go to step 2.

END ALGORITHM 2.

Based on the algorithm 2, if one altered concepts’ relation is larger than the concepts’ previous relation, we may update their CRT structure and move the concepts’ leaves closer. For the example shown in the figure 6, the updated relation between concepts “island” and “Jakarta” is 0.52, which is higher than the their previous relation 0.32 and the relation of “island” and “Indonesia” 0.46. Thus, we need to break the connection between “island” and “Indonesia”, and connect the “island” with the pair of “Jakarta” and “resort” first because they have higher relations. Then, connect “Indonesia” with the new created node that contains “island”, “Jakarta” and “resort.” After CRT is updated,
“island” and “Jakarta” have a lower common ancestor than the concepts “Indonesia” and “Jakarta.”

Figure 6 Updating one CRT based on the altered concepts’ relation

If the CRTs related to the queries $q_1$ and $q_2$ contain some overlapping concepts, we may connect their CRTs by linking the overlapping concepts associating with both of them. Based on CRTs’ similarity, we are able to determine the appropriate weights for their connections. If two CRTs contain lots of overlapping concepts, the CRTs should have strong relationship, so the weight between them should be high; conversely, the weight between them should be low. Thus, we propose the following equation $RTT(CRT_1, CRT_2)$ to calculate the weight between CRTs, in which $n(CRT_1 \cap CRT_2)$ represents the number of concepts occurring in the both CRTs and $n(CRT_1) + n(CRT_2)$ represents the total number of the concepts occurring in $CRT_1$ and $CRT_2$. After the weight between CRTs calculated, we propose the algorithm 2 to connect the overlapping concepts occurring in both CRTs.

$$RTT(CRT_1, CRT_2) = \frac{2 \times n(CRT_1 \cap CRT_2)}{n(CRT_1) + n(CRT_2)}$$

**Algorithm 3** Connecting CRTs

**INPUT:**
$CRT_1$ and $CRT_2$

**OUTPUT:**
A connected CRT that contains $CRT_1$ and $CRT_2$

**BEGIN**
A Personalized Query Suggestion Agent

Step 1: Based on the CRTs relation equation \( RTT(CRT_i, CRT_j) \), calculate the similarity between \( CRT_i \) and \( CRT_j \).

Step 2: Connect \( CRT_i \) and \( CRT_j \) by linking the concepts occurring in both of them. Then, assign the similarity of \( CRT_i \) and \( CRT_j \) to the linkages’ weights.

END ALGORITHM 3.

Given three queries “java”, “holiday tours” and “beverage recipes”, we may construct following CRTs as shown in the figure 6. “Java” is associated with two CRTs, (“tea”, “boca”, “coffee”, and “gourmet”) and (“island”, “Indonesia”, “Jakarta”, and “resort”). “Beverage recipes” is associated with the CRT that contains concepts “tea”, “drink”, “coffee” and “bean.” “Holiday tours” is associated with the CRT that contains concepts “vacation”, “travel”, “Jakarta”, and “Florida.” For the above four CRTs, we find some concepts occurring in more than one CRT, like “tea”, “coffee” and “Jakarta”, so we may connect the CRTs by linking the overlapping concepts “tea”, “coffee” and “Jakarta”. Then, assign the CRTs’ similarities, “0.5” and “0.25” to the linkage weights.

Figure 7 Connecting CRTs based on overlapping concepts

After CRTs are connected, we need to propose the equations for calculating concepts’ relations in the same CRT and in connected CRTs. If \( c_i \) and \( c_j \) are in the same CRT, only one path exists between them, so their relation \( d(c_i, c_j) \) would be their concepts’ relation obtained from the algorithm 1. If \( c_i \) and \( c_j \) are in two connected CRTs \( CRT_i \) and \( CRT_j \), and \( L \) connections between their CRTs, \( c_i \) and \( c_j \) may have \( L \) possible concepts’ relations because one path between them may result in one concepts’ relation. In order to calculate \( L \) possible concepts’ relations between \( c_i \) and \( c_j \), we have to find \( L \) concepts \( \{sc_1, sc_2, ..., sc_L\} \) occurring in both CRTs. Then, we propose the following equation \( CW(c_i, c_j) \) to calculate the relation of concepts \( c_i \) and \( c_j \) in the different CRTs.

\[
CW(c_i, c_j) = \frac{RTT(CRT_i, CRT_j)}{2L} \times \sum_{k=1}^{L} (d(c_i, sc_k) + d(sc_k, c_j))
\]
In the above equation, $s_{ck}$ represents one concept occurring in both $CRT_1$ and $CRT_2$; $d(c_i, s_{ck})$ represents the concepts’ relation of concept $c_i$ and $s_{ck}$ obtained in the algorithm 1; $L$ indicates the number of connections between concepts $c_i$ and $c_j$; $RTT(CRT_1, CRT_2)$ represents the relation between CRTs. Based on the above equation, one relation of $c_i$ and $c_j$ would be the arithmetic average of $d(c_i, s_{ck})$ and $d(s_{ck}, c_j)$ multiplies the weight between $CRT_1$ and $CRT_2$. The relation $CW(c_i, c_j)$ of concepts $c_i$ and $c_j$ in two connected CRTs is the average of all possible relations between $c_i$ and $c_j$.

For example, in the figure 7, we may calculate the relation of concepts “drink” and “boca” by applying the equation $CW(c_i, c_j)$. Two concepts “tea” and “coffee” occur in the both CRTs, so two paths “drink-tea-boca” and “drink-coffee-boca” exist between “drink” and “boca.” For the path “drink-tea-boca”, the relation between “drink” and “boca” should be $RTT(CRT_1, CRT_2)$ multiplying the average of $d(drink, tea)$ and $d(tea, boca)$. As shown in figure 6, the value of $RTT(CRT_1, CRT_2)$ is 0.5, $d(drink, tea) 0.71$ and $d(boca, tea) 0.68$. Then, for the path “drink-tea-boca”, the relation of “drink” and “boca” should be $0.5 \times (0.71 + 0.68)/2$. Similarly, for the other path “drink-coffee-boca”, the relation of “drink” and “boca” should be $0.5 \times (0.41 + 0.47)/2$. The relation $CW(drink, boca)$ should be the average relations for the paths “drink-tea-boca” and “drink-coffee-boca.”

### 2.5 Calculating the Queries’ Relations

Based on the relations between queries and concepts, and relations between concepts and concepts obtained from previous steps, we are able to calculate queries’ relations. If two queries have strong relationship, one of them can be a suggestion for the other.

**Figure 8. Queries – CRTs bipartite graphs**

In the section 2.2, we obtain the concepts related to queries; in the section 2.3 and 2.4, we construct the CRTs based on concepts’ semantic relations and co-occurrence frequencies. After these steps, we can construct the query-CRTs bipartite graph as shown in the figure 8. In the query-CRTs bipartite graph, two queries may connect to the same CRT or different CRTs, and the CRTs may be connected by the concepts occurring in the both CRTs. Based on the above graph, we propose the following two strategies to calculate the queries’ relations.

**Strategy 1.** Two queries $q_i, q_j$ are considered to be related if most of concepts related to $q_i$ have strong relationships with most of concepts related to $q_j$, and $q_i, q_j$ have strong relationships with the related concepts.
A Personalized Query Suggestion Agent

Given two concept sets \(C_i\) and \(C_j\) related to the queries \(q_i\) and \(q_j\) respectively, we propose following three equations \(RQQ1(q_i, q_j), \text{AvgQC}(q_i, C_i), \text{and AvgCC}(C_i, C_j)\) to calculate the relations between \(q_i\) and \(q_j\).

\[
RQQ1(q_i, q_j) = \frac{1}{3} \times (\text{AvgQC}(q_i, C_i) + \text{AvgCC}(C_i, C_j) + \text{AvgQC}(q_j, C_j))
\]

\[
\text{AvgCC}(C_i, C_j) = \frac{1}{mn} \sum_{h=1}^{n} \sum_{k=1}^{m} \text{CW}(c_{ih}, c_{jk})
\]

\[
\text{AvgQC}(q_i, C_i) = \frac{1}{m} \sum_{i=1}^{m} \text{RQC}(q_i, c_i)
\]

In the equation \(RQQ1(q_i, q_j), \text{AvgQC}(q_i, C_i)\), and \(\text{AvgCC}(C_i, C_j)\) represent the average relations between \(q_i\) and the concept set \(C_i\), and \(\text{AvgCC}(C_i, C_j)\) represents the average relations between concept sets \(C_i\) and \(C_j\).

We propose the following equation \(RQQ2(q_i, q_j)\) to calculate the closest relation between \(q_i\) and \(q_j\). \(\text{RQC}(q_i, c_{ih})\) represents the relation between \(q_i\) and \(c_{ih}\), and \(\text{CW}(c_{ih}, c_{jk})\) represents the relation between concept \(c_{ih}\) and concept \(c_{jk}\), which can be obtained in the section 2.4. If \(C_i\) contains \(n\) concepts and \(C_j\) contains \(m\) concepts, there are \(m\) multiplying \(n\) combinations of concepts’ relations between \(C_i\) and \(C_j\). Thus, the equation \(\text{AvgCC}(C_i, C_j)\) represents the average relations between concept sets \(C_i\) and \(C_j\). In the equation \(\text{AvgQC}(q_i, C_i)\), \(\text{RQC}(q_i, c_{ih})\) represents the relation between query \(q_i\) and concept \(c_{ih}\), which can be obtained from the section 2.2.

Strategy 2. Two queries \(q_i, q_j\) are considered to be related if one concept \(c_{ih}\) related to \(q_i\) has a strong relationship with the concept \(c_{jk}\) related to \(q_j\), and \(q_i, q_j\) have strong relationships with \(c_{ih}\) and \(c_{jk}\).

We propose the following equation \(RQQ2(q_i, q_j)\) to calculate the closest relation between \(q_i\) and \(q_j\). \(\text{RQC}(q_i, c_{ih})\) represents the relation between \(q_i\) and \(c_{ih}\), and \(\text{CW}(c_{ih}, c_{jk})\) represents the relation between concept \(c_{ih}\) and concept \(c_{jk}\).

\[
RQQ2(q_i, q_j) = \max(\text{RQC}(q_i, c_{ih}) + \text{CW}(c_{ih}, c_{jk}) + \text{RQC}(c_{jk}, q_j))
\]

Based on the strategy 1, one query can be a suggestion for another query if the concepts related to them have strong relationships. However, if the concepts related to one query are a small sub set of the concepts related to another query, we may not be able to conclude that the first query can be a suggestion for the second one. Thus, we select strategy 2 to calculate the queries relations for query suggestion.

2.6 Dynamically Updating the Weights of Query-Concept

For personalizing query suggestions, the relations between queries and concepts should be dynamically updated based on the users’ most recent clickthrough data. For example, the related concepts for the query “java” may be “code”, “software” and “coffee.” If a user is indeed interested in the concept “coffee” and clicks on the Web pages containing the concept “coffee”, the query suggestions agent should gradually favour the concept “coffee” and the concepts in the same CRT with “coffee”, like “gourmet.” Then, the weight between the query “java” and the concept “coffee” should increase.
The weight between a query and a concept should decrease with the time that the query hits Web pages containing the concept elapses. But the weight should increase if that query hits Web pages containing the concept again. Thus, we propose the following equation $W_Q C(q, c)$ to update the weights between queries and concepts.

$$W_Q C(q, c) = W_Q C(q, c) \times \frac{(1 + \sigma \times R_Q C(q, c))}{(1 + \xi \times \text{elapse time})}$$

In the above equation, the updated weight will be determined by the current weight $W_Q C(q, c)$, the elapse time $\text{elapse time}$, and the relation $R_Q C(q, c)$ obtained from the newest hitting. The initial values for the constant $\xi$ and $\sigma$ are 0.01 and 0.1 respectively.

### 3. Experiments

The experiments and simulations are presented in this section. Firstly, we constructed our test data sets based on the log data of a commercial search engine. Then, we conducted two experiments for evaluating the performance of our method, and compared our method performance with other well-known query suggestion methods.

#### 3.1 Data Collection

We constructed our data sets based on the log data of AOL, a commercial search engine. The log data consist of more than 20M Web queries from 650k users over three months, from March 1, 2006 to May 31, 2006. The number of clicked URLs for the 20M Web queries is 19,442,629. The AOL log data sets can be only used for research purpose only.

For this collection, firstly, we filtered the queries by only keeping the queries that were written in English and only contained alphabet characters and spaces. Secondly, we preserved the queries that resulted in at least five unique clicks per session. Since it was impossible to ask the original users to evaluate the results’ quality for the queries, we assumed the clicked Web pages containing the information that the users needed. Therefore, we used the clicks associated with the queries to approximate relevant Web pages. More relevant documents for a query made it easier to extract concepts related to that query. Thirdly, for better constructing users’ preference profiles, we only kept users’ IDs who submitted more than fifty unique queries. Finally, we randomly selected thirty users’ IDs satisfying above requirements as our test data sets. On average, each user submitted 68.5 distinct queries, and each query resulted in 8 distinct clicks in our data sets. The basic statistic for our data sets is listed in table 1.

<table>
<thead>
<tr>
<th>Item</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Unique Queries</td>
<td>2055</td>
</tr>
<tr>
<td>#Unique User ID's</td>
<td>30</td>
</tr>
<tr>
<td>#Clicked URLs</td>
<td>16440</td>
</tr>
<tr>
<td>#Clicked URLs/ #Queries</td>
<td>8</td>
</tr>
</tbody>
</table>
3.2 Evaluation of Extracting and Clustering Concepts

The process of extracting and clustering concepts aims at extracting important concepts from clicked Web pages and providing representative and distinguishing concepts’ clusters for a given query. We conducted an experiment to evaluate the performance of extracting and clustering concepts. The performance is measured using following aspects:

- Concept, the number of concepts extracted for a given query;
- Redundant concept, the number of not relevant concepts extracted for the query;
- Cluster, the number of concept clusters constructed for the query;
- Redundant cluster, the number of not relevant concept clusters constructed for the query;
- Missing cluster, the number of relevant concept clusters not constructed for the query.

We randomly extracted one hundred non-ambiguous queries and one hundred ambiguous queries from our test data sets. Non-ambiguous query has only one interpretation while ambiguous query contains ambiguous terms and may have multiple interpretations. For the extracted queries, we divided them into ten groups. The first group contained 20 non-ambiguous queries and 0 ambiguous queries. From the second group, each following group contained two more ambiguous queries and two less non-ambiguous queries than the previous group. Therefore, the tenth group contained twenty ambiguous queries and no non-ambiguous queries.

In order to evaluate our algorithm performance, we used the following two well-known algorithms, TF-IDF (Roelleke and Wang, 2008) and DF-ICF (Chen, Xue, and Yu, 2008), for comparison.

TF-IDF is a text mining algorithm that uses the keywords’ occurrence frequencies for concepts extraction. In our experiment, TF-IDF algorithm was used for extracting all concepts related to queries. Then, based on the concepts’ co-occurrence frequencies, we constructed concepts’ clusters.

DF-ICF is a concept clustering algorithm that derives a concept hierarchy from a web directory, and then exploits the semantic knowledge among the concept hierarchy. Based on the concepts’ semantic relations, concepts’ clusters can be constructed. Because DF-ICF cannot extract the concepts from text, we used the concepts gathered from our algorithm as a pre-processing step for DF-ICF.

We selected one group of 10 ambiguous and 10 non-ambiguous queries for comparing the performance of our method, TF-IDF and DF-ICF. The results are shown in the table 2.

<table>
<thead>
<tr>
<th></th>
<th>#Concept</th>
<th>#Redundant Concept</th>
<th>#Cluster</th>
<th>#Redundant Cluster</th>
<th>#Missing cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>186</td>
<td>8</td>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>235</td>
<td>53</td>
<td>14</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>DF-ICF</td>
<td>186</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Yan Chen and Yan-Qing Zhang

We compared those three methods’ performance on our ten groups of queries. The number of redundant and missing clusters created by the methods is shown in the figure 8.

Based on the table 2, we observe that TF-IDF method extracts much more redundant concepts than other methods because TF-IDF method determines the importance of concepts only based on the concepts’ occurrence frequencies, so it cannot filter common meaningless non-stop words such as “conversation”, “information” and “discussion.” Also, as shown in the figure 8, with the growth of ambiguous queries, the number of redundant and missing clusters created by the TF-IDF method grows fast. However, the numbers of redundant clusters created by our method and DF-ICF grow slowly because these two methods use concepts’ semantic relations, not just the concepts’ co-occurrence frequencies, for clustering. Therefore, for a given concept, our method and the DF-ICF algorithm can extract more related concepts to it than the TF-IDF algorithm. Also, we observe that the DF-ICF algorithm misses more relevant clusters than our method because the DF-ICF algorithm determines the concepts’ similarities only based on a predefined category, which may be too coarse and cannot cluster concepts into fine-grained groups that users prefer. Our method calculates concepts’ semantic relations based on a predefined concept hierarchy, WordNet Is-A Trees. Thus, the performance of our method should be at least as well as the DF-ICF algorithm. Moreover, our method uses the concepts’ co-occurrence frequency as a metric for clustering, so more fine-grained concepts’ clusters may be created if the concepts have high co-occurrence frequencies.
3.3 Evaluation of Query Suggestion

In this section, we conducted an experiment to evaluate the performance of query suggestion. Firstly, we randomly selected ten users’ IDs with their submitted queries and clicked URLs from the test data sets, and identified all ambiguous queries and non-ambiguous queries. Then, we divided the submitted queries into ten test cases, and each test case contained more ambiguous queries and less non-ambiguous queries than the previous test case. Secondly, for each unique user, we analyzed the submitted queries and clicked URLs, and best speculated the contents that user wanted to visit. Then, we used the expertise knowledge to manually cluster users’ submitted queries. Thirdly, based on the users’ queries and clicks, we constructed CRTs and query-concept bipartite graphs. Thus, for any query, our algorithm can provide several suggestions based on the weights of query-concept and concept-concept. Because it was impossible to ask the original users to evaluate the quality of our suggestions, we compared them with the queries’ clusters created in the second step. In other words, for a submitted query, we identified one queries’ cluster containing it, and compared the suggestions created by our algorithm with the queries in that cluster. More matched queries between them indicated higher suggestion quality.

We used following three equations to evaluate the quality of query suggestion. The $Q_{relevant}$ is set of query suggestions related to the query, and $Q_{retrieved}$ is set of query suggestions provided by algorithms.

$$ precision(q) = \frac{|Q_{relevant} \cap Q_{retrieved}|}{|Q_{retrieved}|} $$

$$ recall(q) = \frac{|Q_{relevant} \cap Q_{retrieved}|}{|Q_{relevant}|} $$

$$ F = 2 \times \frac{(precision \times recall)}{(precision + recall)} $$

We compared the query suggestion performance of our approach with the following three methods: Adjacency (Huang, Chien, and Oyang, 2003), Query-URL (Beeferman and Berger, 2000) and Query-Concept (Ricardo, Carlos, and Marcelo, 2004) on our test cases. The average precision and recall on our test cases performed by those methods are shown in the table 3. The F-values on the ten test cases performed by those four methods are shown in the figure 9.

Adjacency. Given a query $q_i$, this method lists all queries immediately following $q_i$ in the search history; ranks them based on the frequencies of following $q_i$; and outputs top queries as suggestions.

Query-URL. As discussed in the section 2.1.

Query-Concept. Given a query $q_i$, this method extracts all related concepts from the clicked Web pages; construct Query-Concept bipartite graphs; and output queries linked by the same concepts as suggestions.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>0.657</td>
<td>0.583</td>
</tr>
</tbody>
</table>
Yan Chen and Yan-Qing Zhang

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjacency</td>
<td>0.352</td>
<td>0.257</td>
</tr>
<tr>
<td>Query-URL</td>
<td>0.412</td>
<td>0.403</td>
</tr>
<tr>
<td>Query-Concept</td>
<td>0.532</td>
<td>0.501</td>
</tr>
</tbody>
</table>

**Figure 9** The F-Values performed by our method, adjacency, query-url and query-concept

Based on the table 3, our method obtains highest precision and recall in all four methods. Figure 9 shows that with the growth of ambiguous queries, the F-values of all four algorithms decrease. Overall, the Adjacency algorithm performs worst in all four algorithms because it provides query suggestions only based on the queries’ co-occurrence frequencies. The Query-Concept algorithm performs better than the Query-URL algorithm because fewer queries will be connected by the same URLs than the same concepts. Thus, more suggestions will be provided by the Query-Concept algorithm. Also, as discussed in the section 2.1, the fact of two queries connected by the same URL does not indicate that they must be closely-related queries. However, if two queries are connected by the same concept, they may be closely-related queries. Thus, the performance of the Query-Concept algorithm is better than the Query-URL algorithm. However, if a user submits an ambiguous query, such as “apple”, and the extracted concepts related to “apple” are “fruit” and “computer”, the Query-Concept algorithm cannot justify which concept should be more related to the query. Our method has the ability for identifying the most related concept for the query based on the weights of query-concept pairs. Moreover, because our method defines and updates the concepts’ relations in CRTs, more correct suggestions will be provided by our algorithm than the Query-Concept algorithm.
4. Related Works

Kules et al. (2006) proposed a technique named “Six Fast-Feature” for Web Search Results Clustering, which used predefined features, such as title, snippet, and URL, to classify Web search results into meaningful categories. Wen et al. (2002) proposed a hybrid Web search results clustering algorithm that used both the frequency of clicked pages’ co-occurrence and the similarity of web pages’ contents for classifying Web pages. Based on their methods, two queries are related only if they contain similar terms or they result in the same Web pages clicked. Chen and Choi (2008) defined five top-level genre categories, each of which had several subcategories. Then, Web search results’ categories were identified based on the 31 predefined features, such as the contents, URLs, HTML tags, Java scripts, and VB scripts. Xue et al. (2008) proposed a novel deep-classification approach to categorizing Web documents. The approach first acquired the category candidates for a given document, and then focused the classification effort on the selected category. Thus, only a small subset needed to be classified. However, as discussed in the section 1, because the categories were predefined, classification might be still too general to reflect the accurate granularity aspects of queries.

Carpineto et al. (2001) presented a computationally simple method for Query Expansion. Using ideas from Information Theory, candidate expansion terms were assigned scores. Then, queries were re-written by selecting and weighting those expansion terms. Crabtree et al. (2007) presented a middleware, AbraQ, for Query Expansion. AbraQ first identified the aspects in the query, then identified the aspects underrepresented in the result set of the original query, and finally, for any particularly underrepresented aspect, identified keywords that would enhance that aspect’s representation and automatically expanded the query using the best one. Cui et al. (2003) proposed a framework that extracted probabilistic correlations between query terms and document terms by analyzing query logs. These correlations were then used to select high-quality expansion terms for new queries.

5. Conclusions and Future Works

Because queries submitted to Web Search Engines are usually short and ambiguous, it is difficult for Web Search Engines to speculate the exact meaning of the ambiguous queries. This paper presents a personalized query suggestion agent that uses Query-Concept bipartite graphs and CRTs for query suggestion. Compared to the Query-URL and the Query-Concept algorithms, our method can better identify the queries’ relations and provide more correct suggestions. Also, our agent can dynamically update the weights of Query-Concept and Concept-Concept based on the users’ clickthrough data, and therefore provides personalized suggestions. From the simulation results, our personalized query suggestion agent outperforms other well-known methods.

In future, we will construct a Meta search engine that accepts queries and transfers the queries to traditional search engines. Also, we will deploy our personalized query suggestion agent between the user interface of the Meta search engine and traditional search engines. Thus, the personalized query suggestion agent can collect users’ queries and clicked Web pages, and construct Query-Concept bipartite graphs and CRTs based on these records. Then, the agent can provide personalized suggestions based on the bipartite graphs and CRTs. Based on users’ real selections, we can gather the
Yan Chen and Yan-Qing Zhang

performance of our agent. Moreover, the agent will dynamically update the weights of Query-Concept and Concept-Concept for providing better suggestions.

Reference


**A Personalized Query Suggestion Agent**


