Summer 8-16-2010

Enhanced Web Search Engines with Query-Concept Bipartite Graphs

Yan Chen
Georgia State University

ENHANCED WEB SEARCH ENGINES WITH QUERY-CONCEPT BIPARTITE GRAPHS

by

YAN CHEN

Under the Direction of Yanqing Zhang

ABSTRACT

With rapid growth of information on the Web, Web search engines have gained great momentum for exploiting valuable Web resources. Although keywords-based Web search engines provide relevant search results in response to users' queries, future enhancement is still needed. Three important issues include (1) search results can be diverse because ambiguous keywords in queries can be interpreted to different meanings; (2) indentifying keywords in long queries is difficult for search engines; and (3) generating query-specific Web page summaries is desirable for Web search results' previews. Based on clickthrough data, this thesis proposes a query-concept bipartite graph for representing queries' relations, and applies the queries' relations to applications such as (1) personalized query suggestions, (2) long queries Web searches and (3) query-specific Web page summarization. Experimental results show that query-concept bipartite graphs are useful for performance improvement for the three applications.

INDEX WORDS: Queries, Query-concept bipartite graph, Web search engine, Text mining, Computational intelligence.

ENHANCED WEB SEARCH ENGINES WITH QUERY-CONCEPT BIPARTITE GRAPHS

by

YAN CHEN

A Dissertation Submitted in Partial Fulfillment of the Requirement for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2010

ENHANCED WEB SEARCH ENGINES WITH QUERY-CONCEPT BIPARTITE GRAPHS

by

YAN CHEN

ACKNOWLEDGEMENTS

I sincerely appreciate my advisor, Dr. Yanqing Zhang, for his keen guidance and advice in my research and the defense of this dissertation. His enthusiasm and support make me a better researcher. As an excellent advisor, he is also such a nice person who would like to help students as much as he can.

I would also like to extend my sincere appreciation to Drs. Anu Bourgeois, Vijay Vaishnavi, and Ying Zhu for their invaluable feedback and comments to refine the dissertation.

Above all, I am thankful to my wife Jinxia He for her love, patience, and endless support. Finally, this work is dedicated with deepest love to my parents, Mr. Y.M. Chen and Mrs. H.M. He to whom I stand in debt for my education and knowledge.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 INTRODUCTION

With the exponential growth rate of online data, Web search engines play key roles in retrieving millions of Web documents from the Internet in response to users' queries. Before keyword-based search engines were invented, Internet users might browse Web index sites like Yahoo Indexes for searching their desired information. Because the Web index sites are generally created by human editors and only cover a small portion of Web pages on the Internet, users may not be able to obtain their desired information although the information exists on the Internet. The current keywords-based Web search engines apply the Web Crawler technology to automatically collect and index billion of Web pages so that much more Web pages can be found by Web search engines than by Web index sites. For example, Google indexed almost 8 billion Web pages in 2004 and one trillion pages in 2008 [19]. Because of their simple interfaces and powerful searching abilities, search engines became very popular for internet users. As reported by Daniels, over 90% of web users use search engines to transverse the Web [13]. Currently, search engines are widely used in Web page searches, literature searches, news searches, E-commerce, and online advertisement.

Although Web search engines provide relevant information in response to users' queries, users often experience difficulties in inspecting thousands of search results, spending time for determining relevant results, and summarizing their desired information from inconsistent designed complex pages. Thus, many challenges remain to be solved, which include (1) making searching more accurate and efficient, and (2) discovering, understanding, and summarizing useful knowledge from the search results. The goal of this thesis is to provide users with accurate, effective and personalized results by understanding users' information needs and relevant online text information.

This thesis attempts to solve three research problems, incomplete query Web searches, long query Web searches, and Web search previews. The thesis proposes a new method to explore search queries' relations and apply the relations to solve the above research problems. In the following sections of the Chapter one, the research topics in this thesis are introduced first. Then, the common strategy used for solving the three research problems is presented. Third, the challenges for the research topics in this thesis are discussed, followed by the organization of the thesis.

1.1 Problem Definitions

In this section, three research topics, incomplete query Web searches, long query Web searches, and Web search previews, are introduced respectively. The reasons that they degrade search engine performance are discussed. The general approaches for solving these research problems are presented.

1.1.1 Incomplete query Web searches

Based on the Garrett's conclusions [17], Web searches can be generally divided into two types: Recovery-based search and Discovery-based search. A Recovery-based search is a process of looking for existing online resources. Users know what they need to retrieve from the Internet and they may immediately recognize relevant results returned by search engines. For example, users may want to know the current weather of Atlanta and input "Atlanta weather" to retrieve relevant information. Different from Recovery-based search, a Discovery-based search is a process of exploiting valuable online resources. In other words, users do not have a clear idea of what they want to retrieve online, so they may use descriptive terms to express their information needs and hope to get some clues through the search results. For example, a computer scientist is interested in current hot research topics under cloud computing area. Thus, the scientist may submit a query "cloud computing hot research topics" to Web search engines and get several research topics about cloud computing. Then, the scientist may select on topic he/she is interested in, such as "cloud computing security." Thus, a query used for Discovery-based search can be considered as an incomplete query since the query does not have intact searching requirement.

Another reason that may result in incomplete queries is that users may have clear searching requirements but are unsure of appropriate terms to convey their requirements. Constructing accurate queries depends on users' domain knowledge. If users search in unfamiliar domains, they may use vague or inaccurate terms to specify their requirements and result in a huge number of irrelevant search results [14]. In order to avoid incorrect terminology in queries and irrelevant search results, users may construct queries using a very few keywords only, and then clarify their requirements and refine queries based on the initial searching results.

Because searching incomplete queries may result in diverse results, users have to identify the results meeting their information needs. Therefore, understanding users' information needs on incomplete queries and providing accurate search results became an important research topic for Web search engines.

Providing relevant keywords for the current query based on past search history and asking users to select appropriate keywords to expand queries is an effective way for handling incomplete queries. Formulation of complete queries helps users elicit their search requirements so that search engines may better understand users' information needs. This strategy not only provides appropriate terminology where users do not have enough domain knowledge, but also helps users exploit their interested areas. For example, for one query "car", car owners may be more interested in "car insurance" or "car maintenance" but car dealers may be more interested in "car trader." Thus, providing appropriate keywords to different users based on their searching history is necessary for helping them elicit their search requirements. In this thesis, we proposed personalized query suggestion approach to assist users in completing their queries.

1.1.2 Long query Web searches

Although the typical length of search queries are only two to four terms [46], long queries, as an important query type among the searches on the Web, play a crucial role in many information search applications, such as literature searches, news searches, or searches in environments where queries are expressed as natural language texts but not keywords. The purpose of long query searches is to identify the most relevant information, like articles, news or Web pages, to the contents of long queries. Although long queries are more convenient to express complex and specific information needs than the keyword-based queries, current commercial search engines, in general, perform worse with long queries than with short ones. The major reason is that most search engines respond to a user's query by using the Bag-of-Words model [59], which assumes queries and Web documents are composed of individual words neither related nor ordered. However, for long natural language queries, tokens are ordered words and phrases with underlying semantic relationships. Thus, a lack of sufficient natural language parsing causes search engines not to understand semantic queries [29]. Also, since traditional search engines treat all terms from the users' inputs equally, they lose focus on the key concepts that have the most impact on the retrieval results [3]. Thus, noise or redundant terms will further degrade the effectiveness of long query Web searches if searching all query terms equally [28]. Users may frequently experience the needs to convert their long natural language queries into a few adequate terms for retrieving precise search results, which increases users' burdens.

Choosing a few important words from the original query as the new query is the most common way for improving long query Web searches. Then, search engines may retrieve the most relevant Web pages based on the keywords of the original query and thus avoid irrelevant results. In this thesis, the relevancies of long queries and short queries from browsing history are explored and appropriate short queries that represent the meanings of long queries are selected as substitutions. This approach improves the performance and effectiveness of long query Web searches because search engines may easily identify keywords from the short query substitutions.

1.1.3 Web Search Previews

Because current Web search engines return a large number of search results in response to a query, the query-specific snippets that provide searchers previews of document contents have played an important role for search results presentation [50, 10]. Ideally, snippets should allow users to preview search results' contents and determine a subset of results meeting their information needs. However, because Web search engines generate snippets by extracting query-related short fragments of texts from Web pages, it is difficult for snippets to provide users clear and comprehensive views of search results' contents [57]. Thus, given a specific query, generating query-specific natural language summaries of result pages that aims to minimize users' cognitive loads of accessing Web pages has become an important research topic for Web search engines.

The basic function of query-specific Web page summary is to reveal the major query-related contents for a target Web page and to help a user selecting appropriate pages for their information needs. This thesis proposes a query-specific Web page summarization approach using appropriate queries from browser history to represent the query-related contents in the target Web page.

1.2 Strategy for research problems

Providing incomplete queries with relevant queries from browsing history as suggestions is an effective approach to explore users' information needs. Thus, analyzing relevance between a given query and queries from the history is an important step for query suggestion. As discussed in section 1.1, we also apply queries' relations to improve performance of long query Web searches and Web search previews. Thus, exploring queries' relations is one step for all three applications in this thesis.

Determining queries relevance is a difficult task in the information retrieval area. One reason is that search queries usually contain very few words and traditional text similarity measures cannot use the number of overlap words to determine their relevance [33]. Based on traditional text similarity measures, if queries do not contain any common terms, their relevance scores would be very low although they may express the same meaning. For example, "WTO" and "World Trading Organization" are semantically equivalent, but they do not contain common words. Conversely, although two queries may contain the same terms, they may represent different meanings. For example, "computer mouse" and "Mickey mouse" contain the same term, but they are irrelevant because they refer to different meanings. However, traditional text similarity analysis would suggest that these two queries are very similar because they contain one common term "mouse."

This thesis proposes a query-concept bipartite graph for measuring queries' relations. Compared to the text similarity analysis, this method explores queries' exact meanings through their subsequently clicked Web pages and determines their relevance based on their meanings. Therefore, our approach may measure query relevance more precisely than the text similarity analysis.

1.3 Challenges

Currently, most commercial search engine companies have adopted a lot of innovative algorithms to improve the performance of Web search engines. Also, a lot of academic researchers have proposed diverse approaches to optimize search engines. Different from Web search engineers from commercial companies, academic researchers usually lack enough research data, such as a huge number of searching logs, to verify their approach's performance. Although a few organizations, such as Document Understanding Conference[1] (DUC), have published a number of research data, like queries and descriptions of information needs, it is still not enough to evaluate search engine performance for retrieving billions of Web pages. Because the performance of search engines should be determined by users, researchers have to gather enough people to evaluate the quality of search results by their approach. However, since the evaluation is subjective and depends on users' knowledge and interests, we may get different evaluation results from different users. To evaluate quality of query-specific Web page summaries, we have to ask evaluators to manually create summaries for Web pages. Because different users may construct different query-specific summaries, the evaluation results may vary. In conclusion, we have following challenges in the data collection and evaluation steps.

- Difficult to obtain commercial search engine logs to evaluate algorithm performance.

[1] http://duc.nist.gov/

- Difficult to obtain objective results from users to evaluate qualities of Web searches and query-specific Web page summaries.

1.4 Organizations of the Thesis

Chapter 2 surveys related work on exploring queries' relations and relevant work in incomplete query Web searches, long query Web searches, and query-specific Web page summarization. Chapter 3 presents the query-concept bipartite graphs for exploiting queries' relations. Chapters 4-6 introduce the algorithms that apply the query-concept bipartite graphs for personalized query suggestions, long query reformulation, and query-specific Web page summarization, respectively. Finally, Chapter 7 concludes this dissertation and directs future work.

Chapter 2 RELATED WORK

In this section, the relevant works for exploring queries' relations are introduced first. Because this thesis apply queries' relations for solving three research problems, incomplete query Web searches, long query Web searches, and Web search previews, the related works for these topics are also introduced. For each research problem, the strength and weakness of related works are discussed.

2.1 Related Work for Exploring Queries' relations

As discussed in the section 1.2, exploring queries' relations is a difficult task because queries usually consist of a few words and traditional methods cannot accurately evaluate queries' relations based on very limited contexts. To expand search queries' contexts, Metzler et al. [33] retrieved titles and snippets of top two hundred search results for a given query as its information supplement. Then, they proposed several measures including lexical matching to calculate queries' relations based on their expanded contexts. Sahami and Heilman [40] proposed an approach to expand queries' contexts by collecting keywords from top search results in response to queries. Then, they applied cosine measures to calculate relations of expanded queries. Because these two approaches expand queries' contexts with relevant words, using lexical matching based methods to evaluate queries' relations is possible. However, top search results may not be always relevant to queries. Without users' justifications on results, irrelevant words may be added into queries' contexts and thus original queries' meanings may be obviated. Because users may select relevant results for queries, the exact meanings of queries may be discovered through their subsequently clicked Web pages. This thesis proposes a method to explore queries' relations based on their exact meanings, so accurate queries' relations can be obtained.

Fonseca et al. [15] proposed an association rules-based query relevance evaluation approach. They assumed that Web search queries within the same session had association relations. Thus, the queries' relations can be determined based on the frequency that the queries occurred in the same sessions. However, their approach cannot evaluate queries' relations when those queries belong to different sessions. Gabrilovich and Markovitch [16] applied external sources, such as Wikipedia, to evaluate queries' relations. They represented queries as weighted vectors of Wikipedia-based concepts and then applied cosine similarities to evaluate queries' relations.

2.2 Related Work for Incomplete Query Web Searches

Classifying Web search results in response to incomplete queries is a well-known method for improving performance on incomplete query Web searches. Kules et al. [27] proposed a features-based Web search results clustering approach that classified search results into appropriate categories based on their titles, snippets, and URLs. Wen et al. [55] proposed a Web search results clustering algorithm that used both clicked pages' co-occurrence frequencies and similarities of the pages' contents for classifying Web pages. Their methods considered two queries are related only if they contained similar terms or they resulted in the same Web pages clicked. Xue et al. [58] proposed a domain-based Web documents classification approach. Based on domains for a given document, this approach first obtained its category, and then focused the classification efforts on the selected category. Thus, this approach minimized the number of categories and thus improved the classification performance. The above relevant works have shown that classifying search results is a widely used approach for incomplete query Web searches. However, since 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.

Extending incomplete queries based on relevant terms is another effective way for incomplete query Web searches. Carpineto et al. [6] presented a Query Expansion method that assigned scores to candidate expansion terms based on information theory. Then, terms with high scores were selected and queries were re-written according to the selected terms. Crabtree et al. [11] proposed a middleware for query expansion that identified the terms occurred in the search results but not in the query, and selected appropriate identified terms for expanding original query. Cui et al. [12] proposed a framework that identified 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. Although above approaches expanded semantically related terms to the queries, they did not consider users' preferences, so different users may get the same query extensions although they had different interests.

2.3 Related Work for Long Query Web Searches

Query reduction is a technique that eliminates noisy and redundant terms, and uses underlying retrieving models to extract key concepts from long queries. Formulating a shorter query from the original long query has been proven to lead to significant improvements in Web search applications [28]. Also, searching key concepts from long queries may result in adequate search results since searching long queries has a higher chance of retrieving fewer search results. Kumaran and Allan [28] proposed a query iterative reduction technique that used a small subset of concepts or entities from the original query to substitute the original query and let users determine the most appropriate combinations of term subsets. Their method obtains significant performance improvement on TREC queries. However, this technique is difficult to be implemented due to the exponential number of options that need to be analyzed. Also, this method makes users' burdens heavier. Later on, Kumaran and Allan [29] proposed a selective interactive query reduction approach that determined appropriate query reductions based on users' implicit feedback. The approach reduces users' time and efforts for interacting with the query reductions. Shapiro and Taksa [43] proposed a method that extracted intersecting terms from the long queries as sub-queries and merged the results from searching those sub-queries. Zukerman et al. [60] used decision graphs to identify the noisy attributes of query terms such as syntactic, paraphrase-based and frequency-based attributes from TREC queries, and remove terms with these attributes in long queries. Although query reduction is an effective way for long query Web searches, identifying key concepts in long queries is still a challenge task because of its heavy reliance on corpus statistics. Moreover, since query reduction techniques removes context from the original query, it leads to lose specificity of the original query and to seek more diverse information than the original query.

Query segmentation is a technique that segments queries into concepts, and thus search engines retrieve Web documents based on the concepts but not tokens [18]. Jones et al. [26] proposed a mutual information based approach to determine segmentation breaks between pairs of tokens. Tan and Peng [48] proposed an unsupervised machine learning approach to discover queries' underlying concepts based on a generative language model, and used expectation-maximization (EM) algorithm to estimate the model's parameters. Also, they incorporated Wikipedia to enhance the unsupervised machine learning. Since their approach identifies key concepts of queries, it greatly improves the retrieval performance for long queries. However, since the query segmentation method treats all query concepts equally and loses focus on the key concepts, it degrades the effectiveness of long query Web searches.

Jones et al [28] proposed a query substitution approach that generated a modified query based on past similar queries or phrases, and replaced the original query by the modified query. They built a machine learning model to select the appropriate query candidates based on their features related to the original query and human judgments. Bonchi et al. [4] proposed a query decomposition approach that generated queries whose union of search results roughly matched the results of the original query. They first instantiated the query decomposition problem to the set cover problem, and then applied agglomerative clustering algorithm and dynamic programming to solve the problem. However, their approaches only deal with keywords based short query substitution. Also, replacing one query by relevant queries may lose contexts in the original query, so their approaches may obtain several non-relevant results.

2.4 Related Work for Web search previews

Several summarization approaches that extract sentences containing query terms for constructing query-specific summaries have been proposed. White et al. [57] proposed a sentence scoring scheme through sentences locations in paragraphs, their semantic relations with other sentences, and the proportions of query terms they contain. A number of the highly-scored sentences were chosen to compose the summary. Wei et al. [54] used the mutual reinforcement chains between documents, sentences and terms to select the most representative sentences, and used the query-sentence similarity to measure the affinity between the pair of texts. Otterbacher [35] constructed a graph representation of sentences based on intra-sentences cosine similarity. Then, based on query-sentence lexical similarity, they identified the sentences most likely related to the query. Finally, they recursively retrieved relevant sentences based on the sentence connections on the graph. Although the above methods used different approaches to construct query-specific summaries, all of them attempted to collect salient sentences related to specific queries from the target Web pages as the summaries. We may conclude that these approaches are proposed based on the following two hypotheses.

- H'_1: Sentences that are similar to other sentences in contexts convey important information of Web pages.
- H'_2: Sentences that contain a number of query terms are more likely to convey the information related to the query.

Based on hypotheses H'_1 and H'_2, although we may retrieve sentences containing the major query-related contents of the Web pages, we may also collect several sentences conveying not important or even noisy information since the above hypotheses cannot distinguish the sentences containing query-related topics from the sentences containing common terms that co-occur frequently with the query terms. Thus, summaries created based on the above hypotheses may tend to be long and do not minimize searcher's cognitive load to identify their desired Web pages. Occasionally, the query-related topics may even not occur in target Web pages but require summarizers to underline scatter query-related information and paraphrase them with new sentences. However, the extractive summarization methods do not have such capabilities to analyze and understand the target Web pages' contents and to paraphrase them.

In order to automatically identify query-related topics and contents in target Web pages, Lin and Hovy [31] constructed a topic knowledge base from a predefined corpus and then recognized query-related topics based on the knowledge base during the processes of summarization. However, constructing such a huge knowledge base or corpus that contains any query-related topics is difficult. To identify target Web pages' topics easier, researchers studied clickthrough data and attempted to construct generic Web page summaries based on relations of search queries and their subsequently clicked Web pages.

The previous works [21, 41, 44] indicated that search queries could be used as semantic labels to describe the contents of the subsequently clicked Web pages. Thus, they proposed a query-based text-summarization framework that identified the queries leading to click the target page as labels of its query-related contents and then extracted sentences containing the labels from the page as summaries. Based on Beeferman and Berger's report [2], the probability for two queries lead to one common Web page clicking is less than 10^{-4}. Thus, the above approaches may not retrieve enough queries from the clickthrough data to uncover the major contents of target pages. Also, their approaches cannot construct high-quality query-specific summaries since the queries from the clickthrough data may uncover the major contents for the page but may not uncover contents related to one specific query.

Svore et al. [47] suggested that news query terms or Wikipedia entities are strong signals of user interests so that sentences containing them should be good candidates for news summaries. They proposed a neural network based framework using query terms or Wikipedia entities as features for constructing news article summaries. Hu et al. [22, 23] considered comments contributed by readers providing valuable information to better understand the Web document. They proposed frameworks to extract sentences from Web pages that best represent the topics discussed among its comments. Boydell and Smyth [5] proposed a Web pages summarization technique that used social bookmarking or search queries as basis for summary construction. For a given Web page, they first identified a set of queries that led to the selection of that page among search engine result lists. Second, they submitted the queries to a search engine and collected the query-sensitive snippets from the result sets. Third, they constructed the given Web page's summaries by selecting and combining the snippets obtained in the second step. The above approaches used search queries or other resource as labels of important contents of Web pages for generic Web page summarization, and their experiment results showed significant improvements compared with summarizers without using those labels. However, they did not use these labels in query-specific Web page summarization.

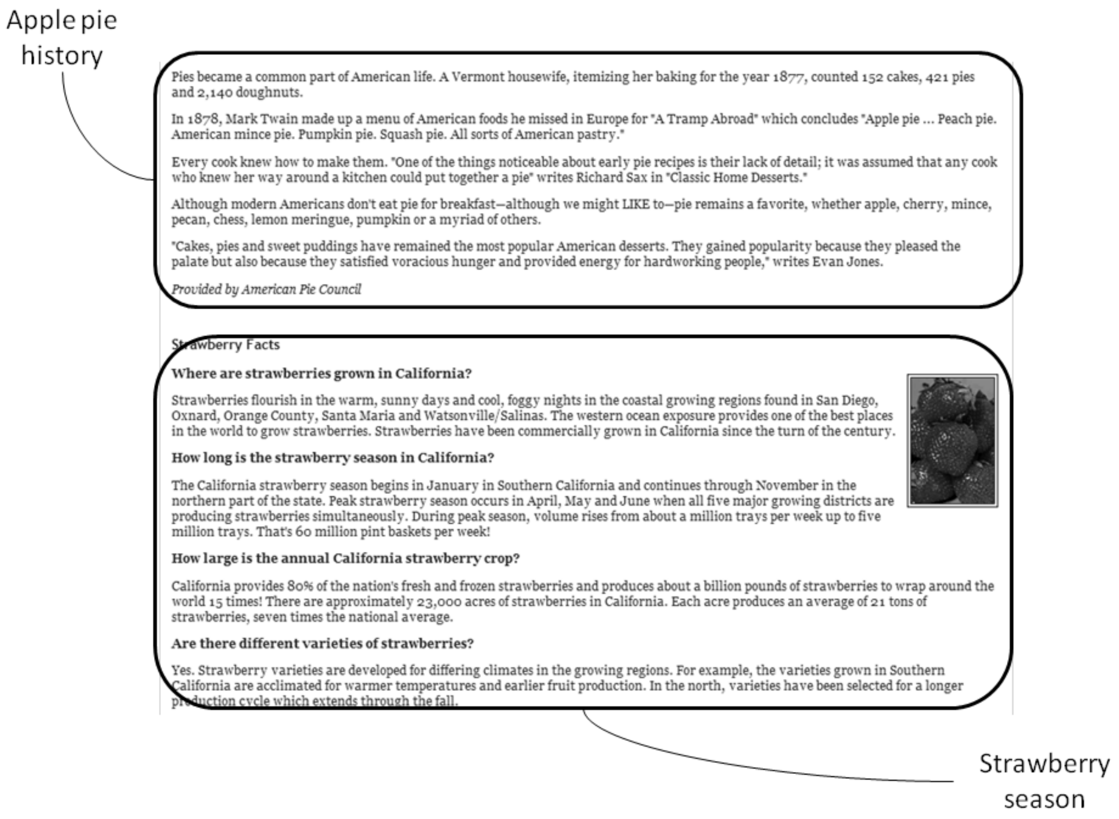
Wang et al. [53] suggested that two documents about one common topic have mutual influence and they would provide information to each other, so they proposed a framework to use mutual information from documents within one appropriate cluster context for collaborative document summarizations. Similar to their strategies, our approach attempts to retrieve queries whose subsequently clicked Web pages containing similar contents as the contents of the target Web page, and used these queries to uncover its query-related topics.

Chapter 3 CONSTRUCTING QUERY-CONCEPT BIPARTITE GRAPHS FOR EXPLORING QUERIES' RELATIONS

This chapter introduces Query-Concept bipartite graphs to investigate queries' relations. In section 3.1, a well-known model for evaluating queries' relevance is introduced, and its strength and weakness are discussed. Section 3.2 introduces the detailed steps to construct Query-Concept bipartite graphs. Two models are compared in section 3.3.

3.1 Query-URL Bipartite Graphs

The clickthrough data that contain users' queries and subsequently clicked URLs are rich resources to investigate queries' relations. Beeferman and Berger [2] believed that two queries were relevant if they resulted in clicking same Web pages. They constructed the Query-URL bipartite graphs to describe the relations of queries and their subsequently clicked URLs. For a given query and its URLs, the queries connected by the same URLs were considered as relevant queries. One major problem for the Query-URL model is that the number of common clicks on URLs for different queries is very 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 was less than 10^{-4} [2]. Therefore, it is very difficult to find the relevant queries for a given query based on the Query-URL bipartite graphs. Another problem with the Query-URL bipartite graphs is that although two queries may lead to same URLs, they may still be irrelevant because they may refer to totally different contents of the Web document. For example, in Figure 3.1, two queries "apple pie history" and "strawberry season" lead to click the same Web page[2] "Food day", but these queries cannot be guaranteed to be relevant because they refer to different contents of "Food day."



Pies became a common part of American life. A Vermont housewife, itemizing her baking for the year 1877, counted 152 cakes, 421 pies and 2,140 doughnuts.

In 1878, Mark Twain made up a menu of American foods he missed in Europe for "A Tramp Abroad" which concludes "Apple pie ... Peach pie. American mince pie. Pumpkin pie. Squash pie. All sorts of American pastry."

Every cook knew how to make them. "One of the things noticeable about early pie recipes is their lack of detail; it was assumed that any cook who knew her way around a kitchen could put together a pie" writes Richard Sax in "Classic Home Desserts."

Although modern Americans don't eat pie for breakfast—although we might LIKE to—pie remains a favorite, whether apple, cherry, mince, pecan, chess, lemon meringue, pumpkin or a myriad of others.

"Cakes, pies and sweet puddings have remained the most popular American desserts. They gained popularity because they pleased the palate but also because they satisfied voracious hunger and provided energy for hardworking people," writes Evan Jones.

Provided by American Pie Council

Strawberry Facts

Where are strawberries grown in California?

Strawberries flourish in the warm, sunny days and cool, foggy nights in the coastal growing regions found in San Diego, Oxnard, Orange County, Santa Maria and Watsonville/Salinas. The western ocean exposure provides one of the best places in the world to grow strawberries. Strawberries have been commercially grown in California since the turn of the century.

How long is the strawberry season in California?

The California strawberry season begins in January in Southern California and continues through November in the northern part of the state. Peak strawberry season occurs in April, May and June when all five major growing districts are producing strawberries simultaneously. During peak season, volume rises from about a million trays per week up to five million trays. That's 60 million pint baskets per week!

How large is the annual California strawberry crop?

California provides 80% of the nation's fresh and frozen strawberries and produces about a billion pounds of strawberries to wrap around the world 15 times! There are approximately 23,000 acres of strawberries in California. Each acre produces an average of 21 tons of strawberries, seven times the national average.

Are there different varieties of strawberries?

Yes. Strawberry varieties are developed for differing climates in the growing regions. For example, the varieties grown in Southern California are acclimated for warmer temperatures and earlier fruit production. In the north, varieties have been selected for a longer production cycle which extends through the fall.

Figure 3.1 Two queries refer to different contents of one Web page

3.2 Query-Concept Bipartite Graph

The Query-Concept Bipartite Graph model is useful to analyze queries' relations even if they do not result in same Web pages. For two given queries, their subsequently clicked Web pages may reveal their exact meanings. If the queries' subsequently clicked Web pages contain very similar contents, the queries are considered to be relevant. We propose the following steps to construct Query-Concept Bipartite Graphs to investigate queries' relations.

A user browses several search results returned by the search engines and locates the information matching his/her query. We assume that if the browsing period for one Web page is long

[2] http://www.globalgourmet.com/food/foodday/fd0197/fd012097.html

enough, that page may contain the information related to the query. Therefore, the following equation *RQD(q, d$_i$)* is proposed for measuring the relevance scores of one Web page d_i with the query *q*.

$$RQD(q,d_i) = \frac{\sum_{j=1}^{m} period_j(d_i)}{\sum_{i=1}^{n}\sum_{j=1}^{m} period_j(d_i)} \qquad (3.1)$$

In the above equation, *m* means that the user clicks the Web page d_i *m* times, and the period of j^{th} visiting is *period$_j$(d$_i$)*; *n* is the total number of clicking Web pages for that query, 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, Web pages that have top relevance scores with the query are selected as the user's favourite Web pages. Then, context terms around the query from the selected Web pages are extracted to express the contents of the Web pages. Based on the assumption that if a keyword or a phrase appears frequently around the query in the user's clicking web pages, it should be an important concept related to the query. The equation *RQC(q, c$_i$)* is proposed to measure the relation between a particular concept c_i and the query *q*.

$$RQC(q,c_i) = \sum_{k=1}^{m} RQD(q,d_k) \times f_{d_k}(c_i) \qquad (3.2)$$

In the above equation, *RQD(q, d$_k$)* is the relation between the query *q* and document d_k containing concept c_i; f_{dk} *(c$_i$)* is the occurrence frequency of the concept c_i in the document d_k; *m* indicates the number of clicked Web pages containing the concept c_i.

After obtaining the relations between concepts and queries, the concepts that have top relevance scores with the given query are extracted to represent the users' favourite information. For example, for the query "java", based on the user's clickthrough data, the related concepts are extracted from Web pages, such as "programming language", "software", "code", "compiler", "technology", "virtual machine", "island", "Indonesia", "Jakarta", "resort", "coffee", "boca", "tea", and "gourmet."

Fig. 3.2 shows a queries- Concept bipartite graph, where squares represent Concepts and rounds represent queries. For any two queries in the bipartite graph, we may speculate their relations based on the common number of context terms they connect.

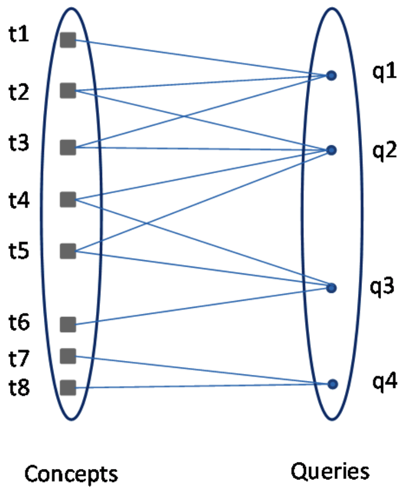


Figure 3.2 A Queries- Concept bipartite graph

3.3 Comparisons of Query-URL Bipartite Graphs and Query-Concept Bipartite Graphs

Both Query-URL bipartite graphs and Query-Concept bipartite graphs are used to investigate clickthrough data for speculating queries' relations. These two models assume that users may select query-related search results from results returned by search engines, and queries' relations can be determined by queries' subsequently clicked results.

As discussed in section 3.1, although two queries may lead to one common Web page clicked, they may be still irrelevant since one Web page may contains multiple topics and the queries are related to different topics in that page. The Query- Concept model retrieves query-related concepts from queries' subsequently clicked Web pages, so the concepts may indicate the query-related contents of the clicked Web pages. Two queries having same concepts from their pages are more likely relevant than two queries resulting in clicking same URLs. Thus, Query-Concept bipartite graphs may get more accurate queries' relations than Query-URL bipartite graphs. Also, the probability that two queries result in clicking same Web pages is very low. The Query-Concept model may retrieve more relevant queries for a given query than the Query-URL model. In summary, compared with Query-URL bipartite graphs, Query-Concept graphs have following advantages:

- The Query-Concept bipartite graphs explore more precise queries' relations than the Query-URL graphs because the latter model ignores clicked pages' contents.
- The Query-Concept graphs get more relevant queries for a given query than the Query-URL graphs because the probability of two queries having the same concepts in their subsequently clicked pages is higher than that of queries resulting in the same URLs clicked.

Chapter 4 QUERIES-CONCEPTS BIPARTITE MODEL FOR PERSONALIZED QUERY SUGGESTION

Query suggestion is a way to extend queries to allow search engines to better speculate exact meanings of incomplete queries. This chapter proposes an approach that uses the Query-Concept bipartite graphs and Concept Relation Trees for personalized query suggestion.

4.1 Methods

This section introduces steps to generate personalized query suggestions for incomplete queries. In step 1, the clickthrough data that contain users' queries and corresponding clicked URLs are analyzed, query-related concepts are extracted, and a Query-Concept bipartite graph is constructed. In step 2, based on concept semantic relations and co-occurrence frequencies, the extracted concepts are clustered, and then Concept Relation Trees (CRT) can be constructed. 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 CRTs, their relations can be calculated based on the relations of Query-Concept and Concept-Concept obtained in the first two steps. In step 3, queries that have strong relationships with the given query are retrieved as suggestions. In step 4, weights of Query-Concept and concepts' relations in the CRTs are updated based on the users' recent queries and clicked URLs. The architecture of the personalized query suggestion agent is shown in Figure 4.1.

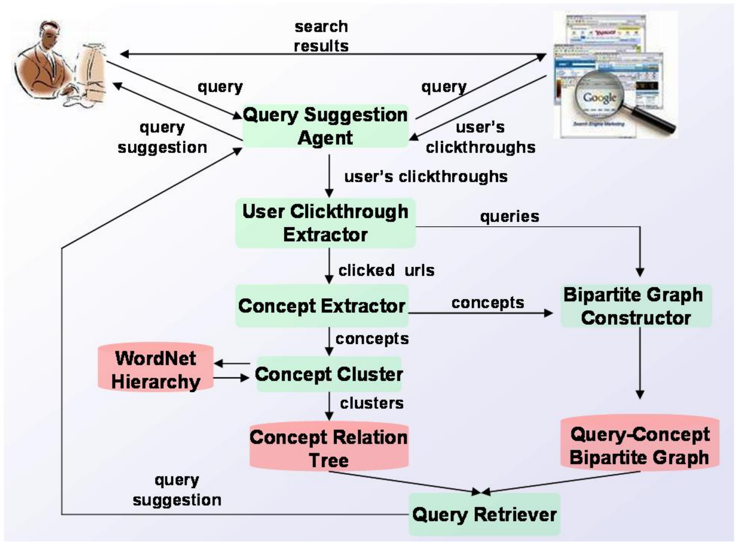


Figure 4.1 The architecture of a personalized query suggestion agent

4.1.1 Constructing queries-concept bipartite graphs

Based on the approaches introduced in section 3.2, all clicked Web pages for a given query are collected and Query-Web page relations are calculated based on equation 3.1. Then, Web pages owning top query relevance scores are selected. Second, the concepts from those Web pages are extracted based on equation 3.2. Thus, we are able to construct a Queries-Concept bipartite graph as shown in Figure 3.2.

4.1.2 Calculating the concepts' relations

After the concepts are extracted from the Web pages, these concepts are divided into different concept sets, and each concept set represents a cluster of closely-related concepts. To construct clusters of extracted concepts, the following equation *RCC(*c_i, c_j*)* is used to calculate the relationship between concept c_i and concept c_j.

$$RCC(c_i, c_j) = \frac{1}{2} \times (SR(c_i, c_j) + CO(c_i, c_j)) \tag{4.1}$$

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 $CO(c_i, c_j)$. In the following parts, the methods to calculate the concepts' semantic relations and co-occurrence frequencies are introduced.

WordNet is a large lexical database in English, developed under the direction of George A. Miller [34]. 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 unique hyponyms in the tree structure. In the Is-A hierarchy tree, each child node is an instance of the parent node, like "car" is instance of "vehicle", and "vehicle" is instance of "physical entity." A part of WordNet Is-A Hierarchy is shown in the figure 4.2.

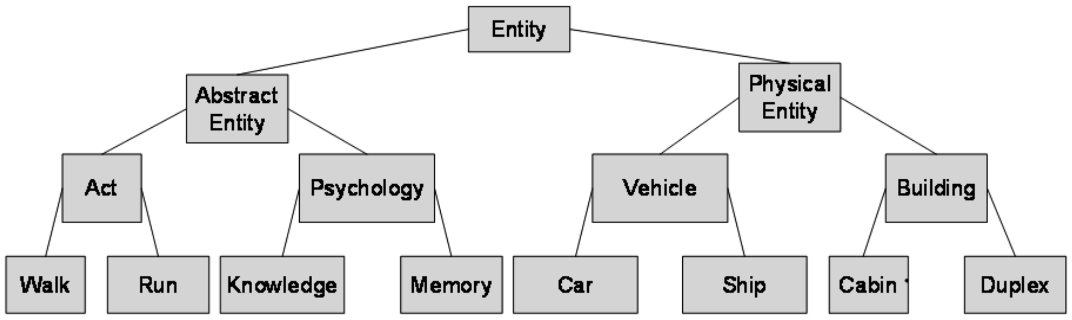


Figure 4.2 A part of WordNet Is-A hierarchy

The semantic similarity between concepts can be estimated by the information content (*IC*) [39]. The information content of a concept *x* is defined as

$$IC(x) = -log(p(x)), \qquad (4.2)$$

where *p(x)* is 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 for a concept is added to its subsuming concept as well. If *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 [49], we propose the following equation $SR(c_i, c_j)$ to calculate any two concepts' relation,

$$SR(c_i,\ c_j) = \frac{|\text{-}log\,\frac{n_p}{n_a}|}{|(\text{-}log\,\frac{n_{ci}}{n_a}) + (\text{-}log\,\frac{n_{cj}}{n_a}) - (\text{-}log\,\frac{n_p}{n_a})|}. \qquad (4.3)$$

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

Based on the above equation, we may estimate that, in Figure 4.1, 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 “physical entity” of “car” and “cabin”, although “car” and “cabin” have the same number of children nodes as the concept “car” and “ship.”

The following equation $CO(c_i, c_j)$ is used to calculate the frequency of co-occurrences of concepts c_i and c_j.

$$CO(c_i, c_j) = \frac{2 \times f(c_i \cap c_j)}{f(c_i) + f(c_j)} \qquad (4.4)$$

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

4.1.3 Constructing concept relation trees

After concepts are extracted based on frequencies of occurring in the users' selected Web pages, concepts with high co-occurrence frequencies and similar semantic relations are grouped together. Based on the concepts' relation equation *RCC*(c_i, c_j), an agglomerative clustering algorithm is developed to construct concept clusters, each of which contains closely-related concepts. Section 4.1.1 extracts relevant concepts for queries. Then, the queries connected by the concepts in same clusters should have strong relationships, and one query of them may be considered as one suggestion of another.

Before presenting Algorithm 4.1, the term "pseudo-concept", a node grouped by the concepts and representing the union of set concepts, is introduced.

In Algorithm 4.1, concepts are grouped together if their relation is larger than a threshold value δ_2. The initial value of δ_2 in Algorithm 4.1 is set as 0.3, and it can be adjusted later. Based on Algorithm 4.1, the extracted concepts are clustered, and CRTs 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. For the previous "Java" example, after the concepts related to the query "java" are extracted and their relations are calculated, based on Algorithm 4.1, following three CRTs are created as shown in Figure 4.3. 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 0.32. Once the CRTs are constructed, they are saved into users' profiles, which can be used as knowledge for personalizing query suggestions.

ALGORITHM 4.1 Constructing CRTs

INPUT:

The extracted concepts, concepts' co-occurrence frequencies and semantic relations

OUTPUT:

Several CRTs, each of which is a concept cluster containing 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. 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 Steps 2 and 3 until all the relations between any concepts or pseudo-concepts are smaller than a threshold value δ_2, or all concepts are grouped into one pseudo-concept.

END ALGORITHM 4.1.

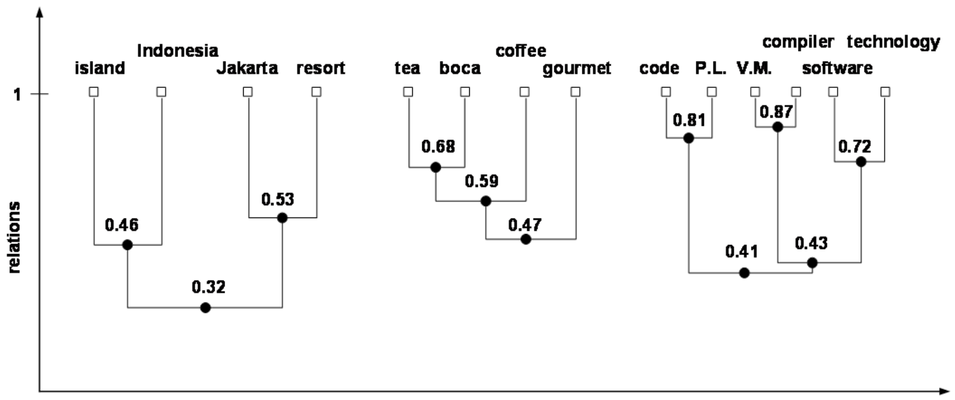


Figure 4.3 Concept relation trees for “java”

For the concepts related to a single query, we can create multiple CRTs, each of which contains closely-related concepts. Also, we may construct separate CRTs related to queries 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, the following approaches are proposed to solve this problem.

If 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. However, we still need to calculate the concepts’ relations for 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. The updated relation for q_2 is the average value of the old relation for q_2 and corresponding concepts’ relation for q_1. For the above “java” example, we construct three CRTs and one of them 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 are updated, the structure of the CRTs needs to be updated to keep concepts’ relations in order. The concepts with lower common ancestors should have closer relations than the concepts with higher common ancestors. Therefore, we propose Algorithm 4.2 to update CRT's structures based on altered concepts' relations.

Based on Algorithm 4.2, if one altered weight is larger than the old weight, their CRT structure is updated, and the concepts' leaves are adjusted closer. For the example shown in Figure 4.4, the updated relation between concepts "island" and "Jakarta" is 0.52, which is larger than the their previous relation 0.32 and the relation of "island" and "Indonesia." 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 larger weights. Then, "Indonesia" is connected with the newly 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."

ALGORITHM 4.2 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 $p^1\ (c_i)$ 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 $p^i\ (c_i)$ or $p^i\ (c_j)$, or both.

Step 4: If the altered relation of concepts (c_i, c_j) is larger than the weight of $p^i\ (c_j)$ but smaller than the weight of $p^i\ (c_i)$, break the connection between the children nodes of $p^i\ (c_j)$. Then, merge $p^i\ (c_i)$ and the child nodes of $p^{i-1}\ (c_j)$. 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 $p^i\ (c_i)$ and $p^i\ (c_j)$, break the connection between the children nodes of $p^i\ (c_i)$ and $p^i\ (c_j)$.Then, merge $p^{i-1}\ (c_i)$ and $p^{i-1}\ (c_j)$.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 and 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 4.2.

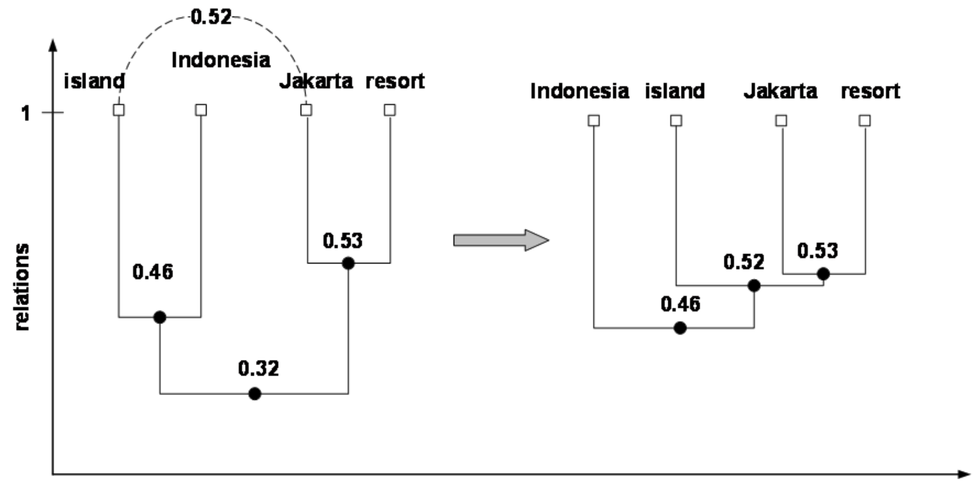


Figure 4.4 Updating one CRT based on the altered concepts' relation

If the CRTs related to the queries q_1 and q_2 contain overlapping concepts, their CRTs are connected by linking the overlapping concepts. If two CRTs contain lots of overlapping concepts, the CRTs should have strong relationship, so the weight between them should be large; conversely, the weight between them should be small. Thus, the following equation $RTT(CRT_1, CRT_2)$ is proposed 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, Algorithm 4.3 is proposed to connect the overlapping concepts occurring in both CRTs.

$$RTT(CRT_1, CRT_2) = \frac{2 \times n(CRT_1 \cap CRT_2)}{n(CRT_1) + n(CRT_2)} \quad (4.5)$$

ALGORITHM 4.3 Connecting CRTs

INPUT:

CRT_1 and CRT_2

OUTPUT:

A connected *CRT* that contains CRT_1 and CRT_2

```
BEGIN
Step 1: Based on the CRTs relation equation $RTT(CRT_1, CRT_2)$, calculate the similarity between $CRT_1$ and $CRT_2$.
Step 2: Connect CRT1 and CRT2 by linking the concepts occurring in both of them. Then, assign the similarity of CRT1 and $CRT_2$ to the linkages' weights.
END ALGORITHM 4.3.
```

Given three queries “java”, “holiday tours” and “beverage recipes”, we may construct following CRTs as shown in Figure 4.5. “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, several concepts occur in more than one CRT such as “tea”, “coffee” and “Jakarta”, so the CRTs can be connected by linking the overlapping concepts “tea”, “coffee” and “Jakarta”. Then, assign the CRTs’ similarities, “0.5” and “0.25” to the linkage weights.

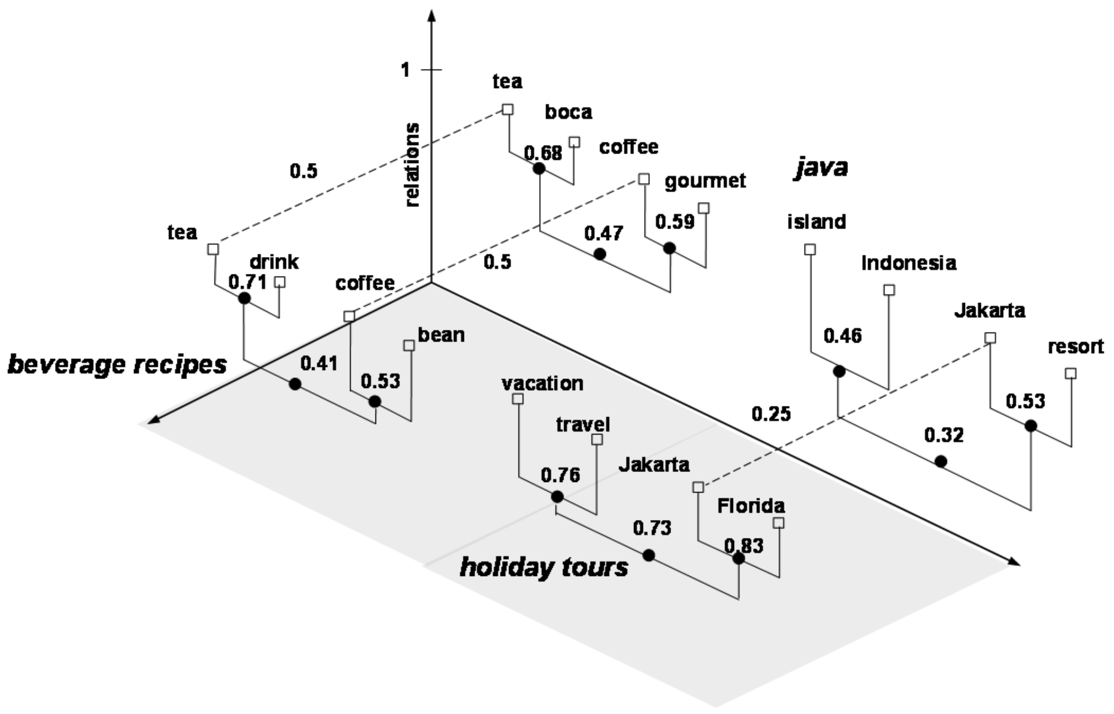


Figure 4.5 Connected CRTs based on overlapping concepts

After CRTs are connected, the equations are needed to calculate 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 $d(c_i, c_j)$ is their concepts' relation obtained from Algorithm 4.1. If c_i and c_j are in two connected CRTs CRT_1 and CRT_2, and L connections exist between 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 $\{sc_1, sc_2, \ldots, sc_L\}$ occurring in both CRTs. Then, the following equation $CW(c_i, c_j)$ is proposed to calculate the relation of concepts c_i and c_j in the different CRTs.

$$CW(c_i, c_j) = \frac{RTT(CRT_1, CRT_2)}{2L} \times \sum_{k=1}^{L} (d(c_i, sc_k) + d(sc_k, c_j)) \quad (4.6)$$

In the above equation, sc_k represents one concept occurring in both CRT_1 and CRT_2; $d(c_i, sc_k)$ represents the concepts' relation of concept c_i and sc_k obtained in algorithm 4.1; L indicates the number of connections between concepts c_i and c_j; and $RTT(CRT_1, CRT_2)$ represents the relation between CRTs.

Based on the above equation, one relation of c_i and c_j is the arithmetic average of $d(c_i, sc_k)$ and $d(sc_k, c_j)$ 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 an average value of all possible relations between c_i and c_j.

For example, in Figure 4.5, the relation of concepts "drink" and "boca" can be calculated 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 4.5, 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" is $0.5*(0.71+0.68)/2$. Similarly, for the other path "drink-coffee-boca", the relation of "drink" and "boca" is $0.5*(0.41+0.47)/2$. The relation *CW(drink, boca)* is the average relations for the paths "drink-tea-boca" and "drink-coffee-boca."

4.1.4 Calculating the queries' relations

Based on the relations between queries and concepts, and relations between concepts and concepts obtained from previous steps, queries' relations can be calculated. If two queries have strong relationship, one of them can be a suggestion for the other. The following two strategies are proposed to calculate the queries' relations.

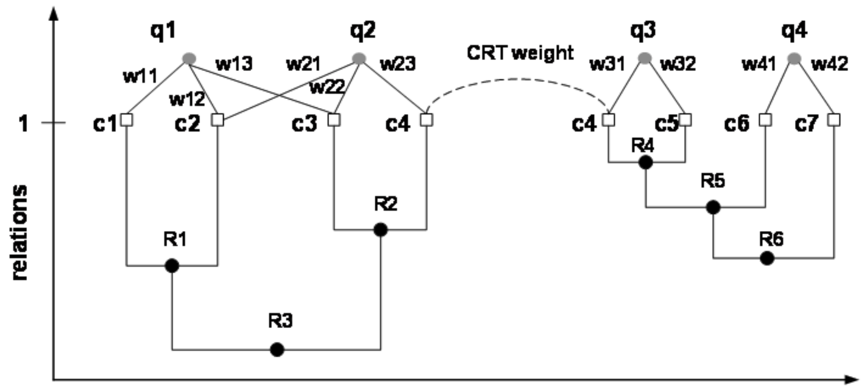


Figure 4.6 Queries – CRTs bipartite graphs

Strategy 4.1. Two queries q_i, q_j are considered to be relevant if most of concepts related to q_i have strong relationships with most of concepts related to q_j.

Given two concept sets C_i and C_j related to the queries q_i and q_j, respectively, the following three equations *RQQ1*(q_i, q_j), *AvgQC*(q_i, C_i), and *AvgCC*(C_i, C_j) are proposed to calculate the relations between q_i and q_j.

$$RQQ1(q_i,q_j)=\frac{1}{3}\times(AvgQC(q_i,C_i)+AvgCC(C_i,C_j)+AvgQC(q_j,C_j)) \quad (4.7)$$

$$AvgCC(C_i,C_j)=\frac{1}{mn}\sum_{h=1}^{n}\sum_{k=1}^{m}CW(c_{ih},c_{jk}) \quad (4.8)$$

$$AvgQC(q_i,C_i)=\frac{1}{m}\sum_{k=1}^{m}RQC(q_i,\, c_k) \quad (4.9)$$

In the equation *RQQ1*(q_i, q_j), *AvgQC*(q_i, C_i) represents the average relations between q_i and the concept set C_i, and *AvgCC*(C_i, C_j) represents the average relations between concept sets C_i and C_j.

In the equation *AvgCC*(C_i, C_j), c_{ih} represents one concept in the concept set C_i and c_{jk} represents one concept in the concept set C_j; *CW* (c_{ih}, c_{jk}) represents the relation between concept c_{ih} and concept c_{jk}. 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 *AvgCC*(C_i, C_j) represents the average relations between concept sets C_i and C_j. In the equation *AvgQC*(q_i, C_i), *RQC*(q_i, c_k) represents the relation between query q_i and concept c_k, which can be obtained from the section 2.2.

Strategy 4.2. 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.

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

$$RQQ2(q_i, q_j) = \max(RQC(q_i, c_{ih}) + CW(c_{ih}, c_{jk}) + RQC(c_{jk}, q_j)) \qquad (4.10)$$

Based on Strategy 4.1, 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, Strategy 4.2 is selected to calculate the queries relations.

4.1.5 Dynamically updating the weights of query-concept

For personalizing query suggestions, the relations between queries and concepts should be updated according to 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 the user 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" is increased.

The weight between a query and a concept is decreased with the time elapses. The weight will be increased if that query hits Web pages containing the concept again. The following equation *WQC*(*q*, *c*) is proposed to update the weights between queries and concepts.

$$WQC(q,c) = WQC(q,c) \times \frac{(1+\sigma \times RQC(q,c))}{(1+\xi \times elapse\_time)} \tag{4.11}$$

In the above equation, the updated weight is determined by the current weight *WQC(q, c)*, the elapse time *elapse_time*, and the relation *RQC(q, c)* obtained from the newest hitting. The initial values for the constant *ξ* and *σ* are 0.01 and 0.1 respectively.

4.2 Experiments

The experiments and performance analysis are presented in this section. First, test data sets were constructed based on the log data of a commercial search engine. Then, two experiments were conducted to evaluate performance of our method.

4.2.1 Data collection

The data sets were constructed 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, first, the queries were filtered by only keeping the queries that only contained alphabet characters and spaces. Second, the queries that resulted in at least five unique clicks per session were preserved. 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. Third, to better construct 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 the data sets. The basic statistic for the data sets is listed in Table 4.1.

Table 4.1 The statistic of the data sets

Item	Statistic
#Unique Queries	2055
#Unique User ID's	30
#Clicked URLs	16440
#Clicked URLs/ #Queries	8

4.2.2 Evaluation of extracting and clustering concepts

The processes of extracting and clustering concepts aim 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 collected one hundred non-ambiguous queries and one hundred ambiguous queries from the 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 [38] and DF-ICF [8], for comparison.

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

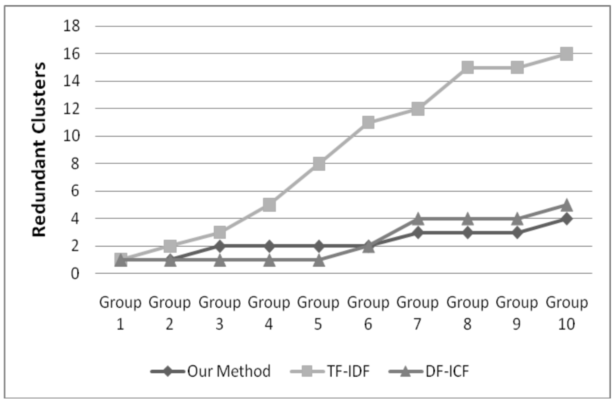
DF-ICF is a concept clustering algorithm that derives a concept hierarchy from a web directory, and then exploits the semantic knowledge among the concept hierarchy. Based on the concepts' semantic relations, concepts' clusters can be constructed. Because DF-ICF cannot extract the concepts from text, 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 to analyze the performance of our method, TF-IDF and DF-ICF. The results are shown in Table 4.2.

Table 4.2 The concepts extraction and clustering performance by our method, TF-IDF and DF-ICF

	#Concept	#Redundant Concept	#Cluster	#Redundant Cluster	#Missing cluster
Our Method	186	8	9	2	2
TF-IDF	235	53	14	8	5
DF-ICF	186	8	7	1	3

We compared those three methods based on the ten groups of queries. The number of redundant and missing clusters created by the methods is shown in Figure 4.7.



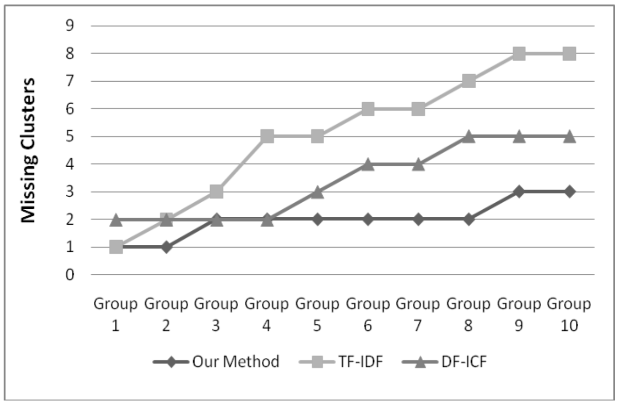


Figure 4.7 The redundant and missing clusters created by our method, TF-IDF and DF-ICF

Based on Table 4.2, TF-IDF method extracts much more redundant concepts than other methods because TF-IDF method determine 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 Figure 4.7, with 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-ICF 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 to 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 comparable with 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.

4.2.3 Evaluation of query suggestion

In this section, we conducted an experiment to evaluate the performance of query suggestion. First, 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. Second, 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. Third, 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.

$$precision(q) = \frac{|Q\_relevant \cap Q\_retrieved|}{|Q\_retrieved|} \tag{4.12}$$

$$recall(q) = \frac{|Q\_relevant \cap Q\_retrieved|}{|Q\_relevant|} \tag{4.13}$$

$$F = 2 \times \frac{(precision \times recall)}{(precision + recall)} \tag{4.14}$$

We compared the query suggestion performance of our approach with the following three methods: Adjacency [24], Query-URL [2] and Query-Concept [37] on our test cases. The average precision and recall on our test cases performed by those methods are shown in Table 4.3. The F-values on the ten test cases performed by those four methods are shown in Figure 4.8.

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 section 3.

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

Table 4.3 The suggestion performance by our method, adjacency, query-url and query-concept

	Precision	Recall
Our method	0.657	0.583
Adjacency	0.352	0.257
Query-URL	0.412	0.403
Query-Concept	0.532	0.501

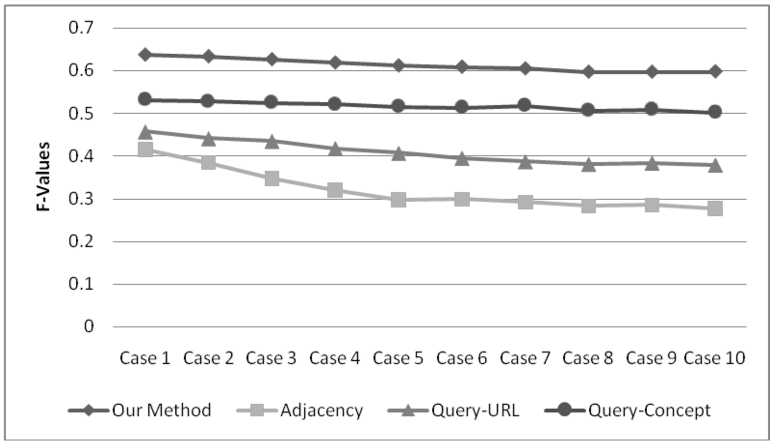


Figure 4.8 The F-Values performed by our method, adjacency, query-url and query-concept

Based on Table 4.3, our method obtains highest precision and recall values in all four methods. Figure 4.8 shows that with 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 are connected by the same URLs than the same concepts. Thus, more suggestions will be provided by the Query-Concept algorithm. Also, as discussed in Section 3.1, 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 are provided by our algorithm than the Query-Concept algorithm.

4.3 Discussion

Since 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 short queries. This section 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 accurate suggestions. Also, our agent can dynamically update the weights of Query-Concept and Concept-Concept based on the users' clickthrough data, and therefore can provide personalized suggestions. From the simulation results, our personalized query suggestion agent outperforms other well-known methods.

The main contributions of our personalized query suggestion approach are:

1. The query suggestion agent uses both concept semantic relations and co-occurrence frequencies for concept clustering. Thus, the clustering performance is higher than the method that uses concept co-occurrence frequencies only.

2. The agent provides more relevant queries as suggestions than other methods.

3. Based on the users' most recent clickthrough data, the agent dynamically updates the weights of Query-Concept and concepts' relations in the CRTs, and therefore adaptively provides the most relevant suggestions for the queries.

Chapter 5 QUERIES-CONCEPTS BIPARTITE GRAPH FOR LONG QUERY REFORMULATIONS

Long queries are widely used in current Web applications, such as literature searches and news searches. However, long queries are frequently expressed as natural language texts but not keywords, so the current keywords-based search engines, like GOOGLE, perform worse with long queries than with short ones. This section proposes a query substitution and search result refinement approach for long query Web searches.

5.1 Methods

Current search engines generally perform worse with long queries than with short keywords-based queries. Searching relevant short queries may provide approximate search results as searching the original long query to improve searching performance. Although the query substitution strategy may improve effectiveness of long query Web searches, diverse results and neighboring information may be still obtained because searching relevant short queries may ignore contexts and terms in the original long query. Thus, we propose the search result refinement strategy that filters non-relevant results by evaluating the similarities of contexts from results and contexts from the original long query.

5.1.1 Substituting a long query by relevant short queries

This section introduces the methods to select relevant short queries for a long query and to build long query substitutions based on these relevant short queries.

The clickthrough data sets that contain users' queries and corresponding clicked URLs are used as data sources to collect relevant short queries. Because users' clickthrough data sets commonly contain a huge number of queries, we need to construct an initial query set *IQ* including a small number of short queries that are potentially related to a given long query. Because long queries may contain multiple sentences, the equation of sentence-query similarity is applied to calculate the relevance score of a long query and a short query [36].

$$R(LQ, SQ) = \sum_{i=1}^{n} \left(\frac{|S_i \cap SQ|}{|SQ|} \right) \tag{5.1}$$

For the above formula, *R(LQ, SQ)* represents the relevance score of the long query *LQ* and the short query *SQ*; S_i denotes the i^{th} sentence in the long query *LQ*; $|S_i \cap SQ|$ denotes the count of common terms occurring in S_i and *SQ*; *|SQ|* represents the number of terms in *SQ*.

Based on the relevance scores of the long query and short queries, the short queries are ranked and *m* short queries are stored in an initial query set $IQ = \{q_1, q_2, \ldots, q_m\}$. Because some short queries contain key concepts of the original long query but others may just contain noisy information, we have to determine which short queries are significant for the long query in the set *IQ*.

For a short query, if a user clicks a corresponding result page, the sentences around the query terms in the clicked result page should contain users' interesting information. Based on this observation, the following two assumptions may be proposed.

Assumption 5.1. The short query's meanings can be discovered from query-related contents in the query's subsequently Web pages.

Assumption 5.2. A short query and the long query are related if the context of the sentences around the short query terms in its clicked result page and the context of the long query are similar.

Based on those assumptions, the following methods are proposed to calculate the contexts' relations. First, the noun phrases are extracted from the set of sentences $ST = \{s_1, s_2, \ldots, s_t\}$ around the short query terms and from the long query. Then, the relations of the contexts of sentences in *ST* and the context of the long query will be determined by the number of common noun phrases they have.

Since the set of sentences *ST* and the long query are natural language texts, we need to determine the noun phrases boundaries such as "(New York) (Art Center)" in the phrase "New York Art Center"; identify name entities such as the names of persons, organizations, and locations; and determine whether two entities refer to each other such as "John Kennedy" and "the president

Kennedy." A machine learning- based noun phrase identification module [56] is applied to recognize fine-grained name entities in the long query and the sentences in *ST*. Also, a corpus-based machine learning approach [45] is adopted for noun phrase co-reference resolution, which resolves a certain type of noun phrase (e.g., pronouns) and also general noun phrases. Once co-referred entities are resolved, for a given referred entity *en*, all other referring entities are replaced by *en* in the long query and the sentences in ST.

Once noun phrases boundaries are identified and co-referred entities are resolved, noun phrases are extracted from the long query and the sentences in *ST* as their features. Then, the following equation is proposed to calculate the context relations of the long query *LQ* and the set of sentences *ST* around the short query *SQ*.

$$R(LQF, STF) = \frac{2 \times |LQF \cap STF|}{|LQF| + |STF|} \qquad (5.2)$$

In equation 5.2, *LQF* denotes the features from the long query; *STF* denotes the features from the sentences in *ST*; *|LQF ∩ STF|* denotes the count of common features occurring in *LQF* and *STF*.

Based on the relations of the long query *LQ* and the set of sentences *ST*, we rank all the short queries in the query set *IQ* obtained in section 5.1.1; select top *n* short queries as the most relevant queries to the original long query; and put top *n* short queries into a relevant query set $RQ = \{q_1, q_2, \ldots, q_n\}$.

Since a long query may contain several topics, its relevant short queries may correspond to different topics of the long query. For example, for a query "Tropical Storm Fay, stalled near Cape Canaveral, Florida, soaked portions of east-central Florida late Wednesday. Florida Gov. Charlie Crist has asked President Bush to declare an emergency in the state to free up federal funding", some relevant short queries may refer to the topic of "Fay stalled Florida", but others may refer to the topic of "governor declare an emergency." Thus, we may construct clusters of queries in *RQ* and select the most representative queries from clusters as the substitutions of the original long query *LQ*.

Since a short query in *RQ* may be included by several sentences in *ST* and each sentence may contain several common features in $|LQF \cap STF|$, we construct a bipartite graph that connects the short queries in *RQ* and the features in *LQF*.

Fig. 5.1 shows a short queries-long query features bipartite graph, where squares represent features and rounds represent queries. Then, we propose the following bipartite agglomerative clustering algorithm [2] to construct short query clusters based on the number of long query's features they connect.

The threshold value δ in the following algorithm is a control factor for determining the sizes of clusters. A small value of δ indicates that the queries in clusters have closer relations, and a large number of clusters may be created; a big value of δ indicates inverse results. In our experiments, we pre-set a reasonable value of δ for obtaining optimal results.

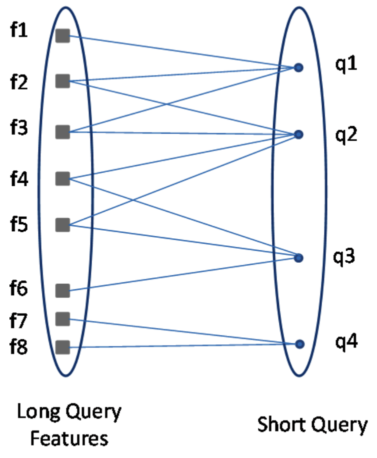


Figure 5.1 A short queries-long query features bipartite graph

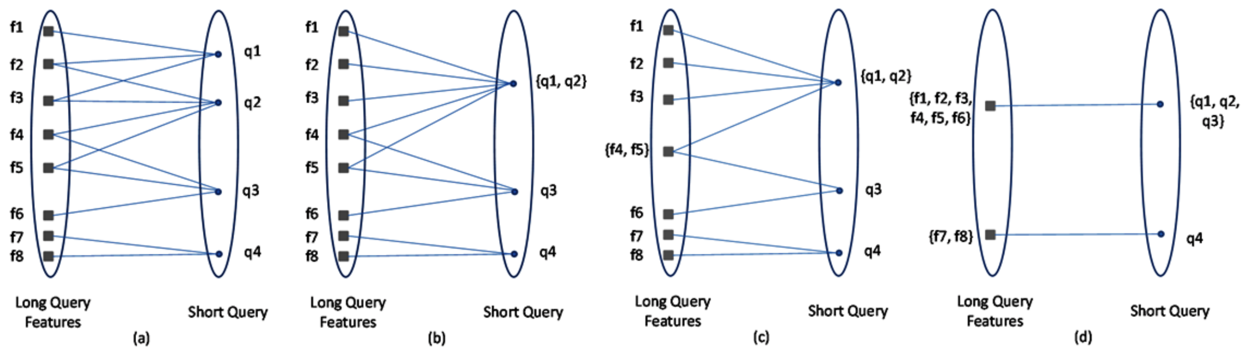


Figure 5.2 A process of constructing clusters of short queries

Algorithm 5.1 – Constructing query clusters

Input: A short queries-long query features bipartite graph G

Output: A clustered short queries-long query features bipartite graph G_c

```
1: Calculate the number of common features between all possible pairs of queries.
2: Merge two queries A and B that connect the biggest number of common features.
3: Calculate the number of common queries between all possible pairs of features.
4: Merge pairs of features connected by the biggest number of common queries.
5: Repeat the steps 1-4 until the number of common features of any pairs of queries is smaller than a
   threshold value $\delta$, or all queries are merged into one query.
```

Fig. 5.2 shows a process of constructing clusters of short queries based on Algorithm 5.1. In Fig. 2 (a), since q_1 and q_2 are connected by the biggest number of common features f_2 and f_3, we merge q_1 and q_2. Then, we connect the new created query $\{q_1, q_2\}$ with the features connected by q_1 or q_2 as shown in the Fig.5.2 (b). Since features f_4 and f_5 are connected by the biggest number of common queries $\{q_1, q_2\}$ and q_3, we merge f_4 and f_5 and connect the new created feature $\{f_4, f_5\}$ with the $\{q_1, q_2\}$ and q_3. We iteratively merge queries and features, and finally obtain two query sets $\{q_1, q_2, q_3\}$ and q_4.

After clusters of relevant short queries are created, we applied the TF-IDF weight [25] to select the most representative query from each cluster and put them together as a new query. Assuming that we have several clusters of queries *C= {c_1, c_2, … , c_m}* and a query *q* from the cluster c_j, we propose the following equation to calculate the query *q*'s score.

$$S(q) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{num_j(t_i)}{\sum_k num_j(t_k)} \times \log\frac{|C|}{|\{c : t_i \in c\}|}\right) \tag{5.3}$$

In equation (5.3), *n* denotes the number of terms that *q* contains; $num_j(t_i)$ denotes the number of occurrence of term t_i in the cluster c_j; $\sum_k num_j\ (t_i)$ is the sum of number of occurrence of all terms in c_j; $|C|$ is the number of clusters; and $|\{c: t_i \in c\}|$ is the number of clusters containing term t_i.

For each cluster, we select the query that has highest score. Then, the selected queries from clusters are combined together as a substitution of the original long query. The newly created query will be submitted to commercial search engines for retrieving relevant documents.

5.1.2 Web pages results refinement

Although the newly created query contains the important features or topics of the original long query, we may still obtain non-relevant results since the new query may ignore a few contexts of the original one. Thus, we propose the following approach that filters non-relevant results by measuring the similarities of search results and the original long query.

First, in the search results, we identify the sentences containing the terms of the newly created query. Second, we apply the machine learning-based noun phrase identification module used in section 5.1.1 to recognize entities or noun phrases in these sentences. Third, we propose equation 5.4 to calculate the relevance scores of search results and the original query. Then, the search results will be ranked based on the relevance scores and only top results will be returned to users.

$$S(p_i) = \sum_{j=1}^{n}\left(\frac{|f_i \cap F(t_j)|}{|F(t_j)|}\right)^{\frac{1}{k}} \tag{5.4}$$

In equation (5.4), p_i denotes the i^{th} search result; f_i denotes the features extracted from p_i; t_j denotes the *j*'s topic; $F(t_j)$ denotes the features related to topic t_j in the original long query; and constant *k* is a control factor to determine the impact of relatedness of features f_i and $F(t_j)$. As *k* increases, the p_i may obtain high scores if several common features exist between f_i and $F(t_j)$.

5.2 Experiments

The experiments are presented in this section. First, we constructed our data sets based on the TREC Robust 2004 data and three month search log that we used in Chapter 4. Then, we compared our approach with other two well-known approaches.

5.2.1 Data collection

We constructed the long query test sets based on the TREC Robust 2004 test file, which contained 250 titles, descriptions and narrative sections. Since the descriptions are natural language texts and express the information needs, the descriptions of TREC Robust 2004 are considered as the long queries in the experiments. Since the titles express the same information needs as the descriptions and they commonly contain a few keywords, the titles are considered as short queries in the test sets. In the experiments, titles are considered as baseline to compare with our approach. The lengthy narrative sections express important attributes of search results. Thus, we may consider the narrative sections as measures to evaluate whether retrieved documents meet the information needs. Since the descriptions of TREC Robust 2004 always contain one topic, we selected 100 descriptions of TREC Robust 2004 as the single-topic long queries. Also, we manually created 100 multi-topics long queries by combining the descriptions with related topics from the narratives sections of TREC Robust 2004. We constructed two sets of long query testing data. The first test set contained all single-topic long queries. The second test set contained all single-topic and multi-topic long queries. The queries in second test set were divided to ten groups. The first group contained 20 single-topic queries and 0 multi-topic queries. From the second group, each following group contained two more multi-topic queries and two less single-topic queries than the previous group. Therefore, the tenth group contained twenty multi-topic queries and no single-topic queries.

We constructed our short query data sets based on the log data that were used in Chapter 4. For the query collection, first, we filtered the queries by only keeping the queries written in English and only containing alphabet characters and spaces. Second, we only preserved the queries that contain less than three terms as short queries. Third, we preserved the queries that resulted in at least one click per session. Since it was impossible to ask the original users to evaluate the search results' quality, we assumed that all the clicked Web pages containing the information that the users needed. Based on Assumption 5.1, the queries should represent the key contents of the sentences around the query terms in the clicked result pages. Then, for our long query test data, equations 5.1 and 5.2 are used to retrieve no more than twenty relevant short queries from the short query sets as the working sets.

5.2.2 Experiment design

Parameters Setting. Since our initial experiments showed that a large set of short query candidates may contain noisy information irrelevant to the original long query, we selected no more than five relevant short queries for a long query.

Since the test sets contained one hundred single-topic long queries and one hundred multi-topic long queries, and the average number of topics of long query test set was 1.7, we selected a reasonable value of parameter δ to construct clusters of relevant queries in algorithm 5.1. Based on our initial experiments results, the value range of δ should be between 1 and 3 so that we may create an approximate number of clusters as the number of topics in the test sets.

In order to enlarge the impact of relatedness of features extracted from result pages and features from the original long query, we used *k=2* in equation 5.4.

Comparative Approaches. We used the titles of TREC Robust as baselines to analyze the performance of our approach. Also, we compared our approach with the following two well-known methods.

Concept Weighting. Bendersky and Croft [3] proposed a machine learning-based concepts weighting approach to identify key concepts in verbose queries. They constructed vectors containing seven weighting features to represent concepts. Then, for the training set, they seek to learn a ranking function so that the rank of concept *i* is higher than concept *j* if the function value of concept *i*'s feature vector is larger than concept *j*'s. Once the learning process finished, key concepts were selected from verbose queries based on their ranks.

Query Reduction. Shapiro and Taksa [43] proposed an approach that contained following steps for query reduction: removing stop words from the original query and decomposing the original queries into several search terms; ranking those search terms based on the TF-IDF Weight; generating a fixed/variable number of sub-queries by randomly selecting terms from the search terms; submitting sub-queries as inputs to a search engine and merging the search results from those sub-queries into one rank list.

Evaluation Measures. Two measures, P@10 and MAP, are used to compare our approach with others. P@10 is the precision of top 10 rank documents returned by a search engine. MAP [9] is the arithmetic mean of average precisions (AP) of a set of queries and AP is the sum of the precision at each relevant document in the search results divided by the total number of relevant documents in the collection.

Correct Results for Testing Data. Since we did not have a standard result set to evaluate the correctness of the search results of long query Web searches, we had to manually speculate the correctness of search results. In order to minimize the bias for the results speculation, we collected the search results from all approaches; constructed indexes for search results and put indexes in a separate file; put all results in a corpus and justified their correctness. Thus, before justifying search results, we did not know which approach creates them, so this approach minimizes justification bias.

5.2.3 Experiment results

Our main hypothesis is that substituting a long query by relevant short queries and filtering non-relevant search results are more beneficial than machine learning-based key concepts identification or query reduction. Also, compared with other approaches, our approach's performance is less impacted by the number of multi-topic long queries. Thus, in the following experiments, we compare our approach with other approaches on the test sets one and two.

Table 5.1 shows the comparison results of three approaches in terms of P@10 and MAP on the test set one. Baseline approach and the query reduction approach perform worse than concepts weighting and our approach, and our approach performance is slightly better than the concept weighting. In the baseline approach, queries are comprised of a few keywords, and they do not contain enough contexts to describe the information needs, so the search results may be diverse. The query reduction approach creates sub-queries by randomly selecting terms from the original query, so a lot of non-meaningful or noisy queries may be created. Thus, searching those sub-queries in parallel and combining their results may obtain numerous non-relevant results and greatly hurt the searching precision. Although the concept weighting approach achieves better performance than query reduction and baseline approaches, its performance is heavily relied on corpus statistics. Since our approach selects the most representative queries to substitute the original long query, and filter non-relevant results, our approach significantly improve the searching performance.

Table 5.1 Comparison of different methods by P@10 and MAP on test set 1

Method	Test Set 1	
	P@10	MAP
Baseline	0.235	0.209
Query Reduction	0.218	0.184
Concepts Weighting	0.314	0.281
Query Substitution	0.334	0.317

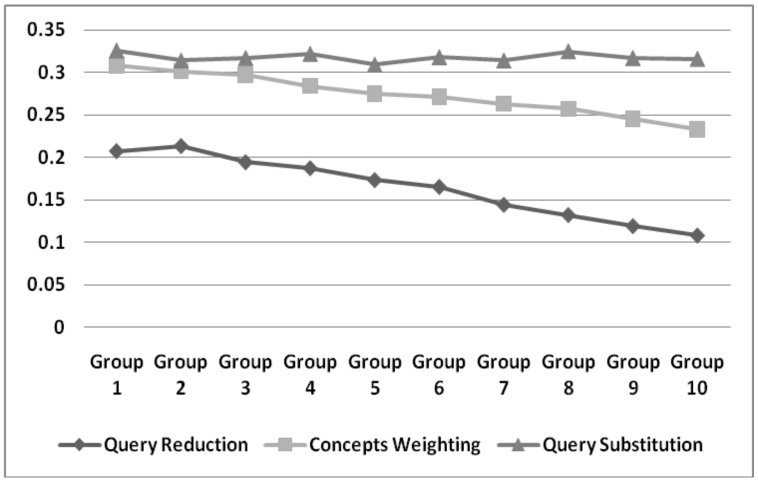


Figure 5.3 Comparison of different methods by P@10 on Test Set 2

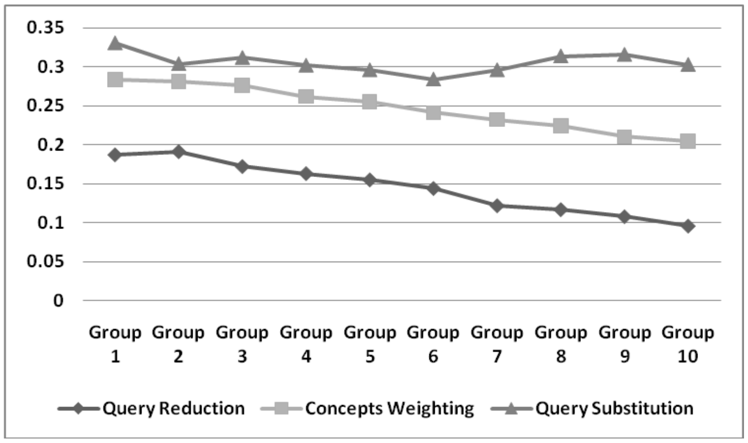


Figure 5.4 Comparison of different methods by MAP on Test Set 2

We still used P@10 and MAP as evaluation metrics to compare three methods on the second test set. Since the second test set contains ten groups of long queries, and the latter group contains more multi-topic queries than the previous group, we showed all approaches' performance on these ten groups of test data in Fig. 5.3 and Fig. 5.4. With the number of multi-topic queries increases, the trends of performance of these approaches can be observed.

As shown in Fig. 5.3 and Fig. 5.4, with growth of the number of multi-topics queries, the performance of the query reduction and concepts weighting approaches drops fast while our approach's performance is not heavily impacted. Among three approaches, the performance of query reduction drops fastest since it randomly selects terms from the original query to create new sub-queries. With the number of topics increases in the query, more non-meaningful or noisy sub-queries will be created, and thus more non-relevant results will be retrieved. Since the concepts weighting approach identifies key concepts in single-topic queries instead of multi-topic queries, its multi-topic long query performance is not as good as the single-topic long query performance. Since our approach constructs clusters of short queries based on the topics that they refer to, and selects the most representative query from each cluster, our approach is less impacted by the multi-topic queries compared to other methods.

5.3 Discussion

This section proposes a query substitution-search result refinement approach that uses relevant short queries to replace the long query and then filter the non-relevant queries based on the context matching. The experiments prove that this approach performs better than concepts weighting and query reduction approaches. Also, for the multi-topic queries, this approach is less impacted than the above two approaches.

The major contributions of this paper are:

1. Propose a Query Substitution – Search Result Refinement approach to improve the performance of long query Web searches.
2. For the long queries containing several topics, this approach significantly improves the performance of Web searches.

Chapter 6 QUERIES-CONCEPTS BIPARTITE GRAPH FOR QUERY-SPECIFIC WEB PAGE SUMMARIZATION

This chapter proposes a query-specific Web page summarization approach that retrieves relevant past queries to uncover the query-related contents of target Web pages and constructs query-specific Web page summaries based on these retrieved queries.

6.1 Methods

This section introduces steps to generate query-specific Web page summaries. For a given query, we locate the sentences containing query terms from the target Web page; identify context boundaries for the selected sentences; and extract context terms around the query terms. In the second step, we browse searching log data and retrieve past queries that have lexical similarities with the given query and its context terms as the candidate labels of the query-related topics of the target page. Since we may retrieve some irrelevant candidate labels for the target Web page, in the third step, we filter those irrelevant labels. We extract context terms around candidate labels from their subsequently clicked Web pages; compare them with context terms around the given query from its subsequently clicked Web pages; and remove labels from the candidate pool if their contexts are not similar to the given query's contexts. Also, for a target Web page with multiple query-related topics, we may retrieve queries related to different topics. Thus, we construct label clusters based on similarities of their around context terms from their subsequently clicked Web pages with terms around the given query from the target Web page. Consequently, we may obtain label clusters to uncover query-related topics and one cluster may correspond to one topic. Next, we select the most representative label from one cluster and extract sentences containing the label's terms from the target Web page as summaries of that query-related topic. If we cannot locate appropriate sentences containing the labels in the target Web page, the page may use different words conveying the same meanings, so we extract sentences from the target Web page that contain the context terms around labels as summaries.

Most current Web page summarizers attempt to construct query-specific summaries for all Web pages. Based on a search engine user behavior study [42], 62% of search users only click Web links on the first search result page and 90% of users only click links on the first three result pages. Thus, this chapter attempts to construct query-specific summaries for Web pages on the first three result pages.

In the following sections, we introduce our approach for constructing query-specific Web page summaries based on relevant search queries. Section 6.1.1 presents the methods for retrieving query-related context terms from the target Web page. Section 6.1.2 retrieves past relevant queries as the candidate labels for uncovering the query-related topics of the target page. Then, our approach filters irrelevant labels and constructs label clusters corresponding to the query-related topics. Section 6.1.3 extracts appropriate sentences from the target Web page as the query-specific Web page summaries.

6.1.1 Retrieving query-related context terms from the target Web page

Tombros and Sanderson [50] concluded that the sentences containing more query words more likely express important information related to the query. In order to identify the important context words related to a specific query, sentences containing enough query terms are located. The following sentence scoring formula is used to calculate the relevance scores of sentences for a given query: [10]

$$\omega(x) = \frac{1}{|x|}\sum_{t \in T} x(t) \qquad (6.1)$$

where *|x|* is the number of distinct terms sentence *x* contains; *T* is the universal set of all terms used in the query *q*; and *x(t)* = 1 if the sentence *x* contains the term *t* and 0 otherwise.

Based on the relevance scores of sentences, we rank the sentences from the source Web page, collect top *m* sentences, and build a set of selected sentences $ST = \{s_1, s_2, \ldots, s_m\}$. Once those query-related sentences are located, the query-related context terms can be extracted from those sentences.

However, since the around contexts of selected sentences may also contain the query-related terms, context boundaries of the selected sentences are identified for better extracting query-related terms.

Based on Varadarajan and Hristidis suggestions [19], we attempt to identify the context boundaries only inside the paragraphs containing the sentences. Then, we applied TextTiling [20] for detecting paragraphs subtopics shift and thus segmenting paragraphs. First, this approach calculates the sentences' similarity scores by the following equation,

$$score(i) = \frac{\sum_t w_{t,st_i} w_{t,st_{i+1}}}{\sqrt{\sum_t w^2_{t,st_i} \sum_t w^2_{t,st_{i+1}}}} \qquad (6.2),$$

where *score(i)* represents the similarity score between sentences s_i and s_{i+1}; t represents the terms from the paragraph; st_i represents the terms from s_i; and $w_{t,st}$ is the weight assigned to term t in the sentence s_i, which is term t's frequency in s_i. Since similarity scores of adjacent sentences are cues of topic shifting, we rank the similarity scores and select the gaps between sentences that have low scores. Then, for sentences selected based on equation 6.1, we identify their around gaps as the context boundaries for them.

After context boundaries for query-related sentences are identified, query-related context terms are extracted from the sentence contexts. [50, 54] concluded that the frequency of terms co-occurring with the query terms is one important factor to determine their significance to the query. Thus, we rank terms inside of query-related contexts and extract top rank terms as query-related context terms.

6.1.2 Retrieving relevant queries as labels to uncover query-related contents

In Section 6.1.1, context terms for a given query are retrieved from the target page, some of which may indicate query-related topics of that page while others may be normal terms and do not contain valuable information related to the query. The human editors may identify the query-related topics from the context terms and thus construct summaries based on those topics. However, without external knowledge, it is difficult for machines to automatically identify correct terms as query-related topics.

For a given query, its top ranking Web search results may contain popular query-related contents. Because of their popularity, some other relevant queries may be proposed for searching the similar contents also. Thus, we made the following hypothesis one.

- H_1: The query-related topics from its top ranking Web search results may be described by other relevant queries.

Since we construct query-specific summaries only for Web pages on the first three result pages, appropriate queries that attempt to search similar contents as the given query may exist in the searching history.

Because queries are created by users and for expressing their search needs, those queries may use a few keywords to convey the meanings of contents. Therefore, if one past query is strongly related to the current query and its context terms in the target Web page, the query may contain appropriate keywords for describing the relevant contents in the top ranking Web pages.

In this section, we extract search queries strongly related to the given one and it's around context terms from the target Web page as candidate labels to describe the contents. Based on the *Term Frequency Weight* [25], the following equation 6.3 is proposed to calculate relevance of past queries with the given one,

$$rel(c \mid q,t) = \sum_{w \in q} \log(tf_{w,c}+1) \times (\log(tf_{w,q}+1) + \sigma \times \log(tf_{w,t}+1)) \qquad (6.3)$$

where $tf_{w,c}$, and $tf_{w,q}$ are the frequencies of term w occurring in a past query c and the given query q, and $tf_{w,t}$ the frequencies of term w co-occurring with a context term t. σ is a control factor to determine the impact of $tf_{w,t}$ for the relevance scores. Based on the relevance of past queries and the current query, we select top relevant past queries as candidate labels for describing the query-related topics of the target Web page.

As discussed in the introduction part, not all labels refer to query-related contents of the page, so we need to identify and filter irrelevant labels from the candidate pools. Based on the Sun et al. suggestions [44] mentioned in the introduction section, the hypothesis two is made to identify the exact meanings of queries.

- H_2: The meanings of queries may be described by the context around them in their subsequently clicked pages.

Based on the hypothesis two, we may distinguish relevant and irrelevant labels by comparing their around context terms in their subsequently clicked pages with the context terms around the given query in the target Web page. Strong similarity between terms around one label and the terms around the given query may suggest that label is relevant to the query-related contents of the target page. Thus, we apply the steps used in section 6.1.1 to extract the context terms around labels from their subsequently clicked pages. Then, we build the following Contexts -Context Terms-Labels tripartite graph, where Contexts are extracted from the target page and contain the given query terms, Context Terms are important terms inside the contexts and around the query terms. Both of them can be obtained in the section 6.1.1.

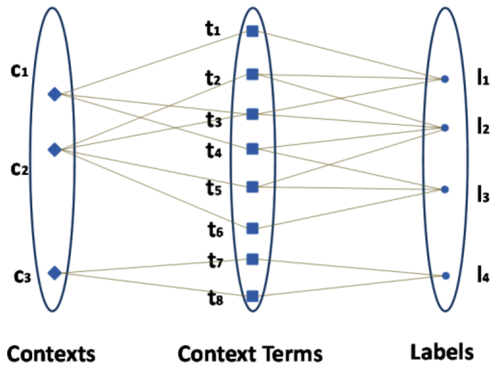


Figure 6.1 A Contexts-Context Terms-Labels tripartite graph

Figure 6.1 shows a Contexts-Context Terms-Labels tripartite graph, where squares represent context terms, diamonds represent contexts and rounds represent labels. A connection between one term and one label indicates that term occurring in the target page and the label's subsequently clicked pages. A connection between one term and one context indicates the context containing the term. Based on the number of common context terms around labels and the given query, we propose the algorithm one to construct clusters of labels. After clustering, label clusters will uncover query-related topics of the target page and irrelevant labels will be removed. Also, the algorithm one constructs clusters of contexts and context terms so that the target page's contents referred by label clusters can be identified.

The c_l and c_t scores in the algorithm one are two metrics to evaluate label clusters' similarities and context term clusters' similarities respectively. They are defined in equations 6.4 and 6.5.

$$c_l = \frac{2 \times |LC_1 \cap LC_2|}{|LC_1| + |LC_2|} \quad (6.4), \quad c_t = \frac{2 \times |TC_1 \cap TC_2|}{|TC_1| + |TC_2|} \quad (6.5).$$

In equation 6.4, LC_1 and LC_2 denote the label cluster one and two; $|LC_1 \cap LC_2|$ denotes the number of common labels in both of them. TC_1 denotes one term cluster in equation 6.5.

The threshold value δ in the algorithm one is a control factor for determining the sizes of label clusters. A small value of δ suggests that the labels in clusters have strong relations and a large number of clusters may be created; a big value of δ suggests inverse results. In our experiments, we pre-set a reasonable value of δ for obtaining optimal results.

Figure 6.2 shows a process of constructing clusters of labels based on the algorithm one. In Figure 6.2 (a), since l_1 and l_2 are connected by the largest number of common context terms, we merge l_1 and l_2. Then, we connect the new created label $\{l_1, l_2\}$ with the context terms that were connected by l_1 or l_2 respectively as shown in Figure 6.2 (b). Since context terms t_4 and t_5 are connected by the largest number of common labels, we merge t_4 and t_5 and connect the new created term $\{t_4, t_5\}$ with the $\{l_1, l_2\}$

and l_3 in Figure 6.2 (c). In Figure 6.2 (d), since {l_1, l_1} and l_3 are connected by the largest number of common context terms, we merge {l_1, l_1} and l_3. Based on the algorithm one, we iteratively merge labels and context terms. Finally, we obtain two label sets {l_1, l_2, l_3} and l_4 and their corresponding term sets {t_1, t_2, t_3, t_4, t_5, t_6} and {t_7, t_8} as shown in Figure 6.2 (e). Then, we merge contexts so that context terms connected by one label are in the same context.

Algorithm 6.1 – Constructing clusters of labels

Input: A Contexts-Context Terms-Labels tripartite graph G

Output: A clustered Contexts-Context Terms-Labels tripartite graph G_c

```
1: If the $c_l$ scores between all pairs of labels are smaller than a threshold value $\delta$, or all labels are in one cluster, go to step 4; otherwise, go to step 2.
2: Calculate the $c_l$ scores between all possible pairs of labels. Merge two labels $A$ and $B$ that have the largest $c_l$ scores, and connect their corresponding terms to the new created label.
3: Calculate the $c_t$ scores between all pairs of context terms. Merge two terms $a$ and $b$ that have the largest $c_t$ scores, and connect their corresponding labels and contexts to the new created term. Go to step 1.
4: Repeat merge contexts until the all context terms connected by the same label are in one context.
```

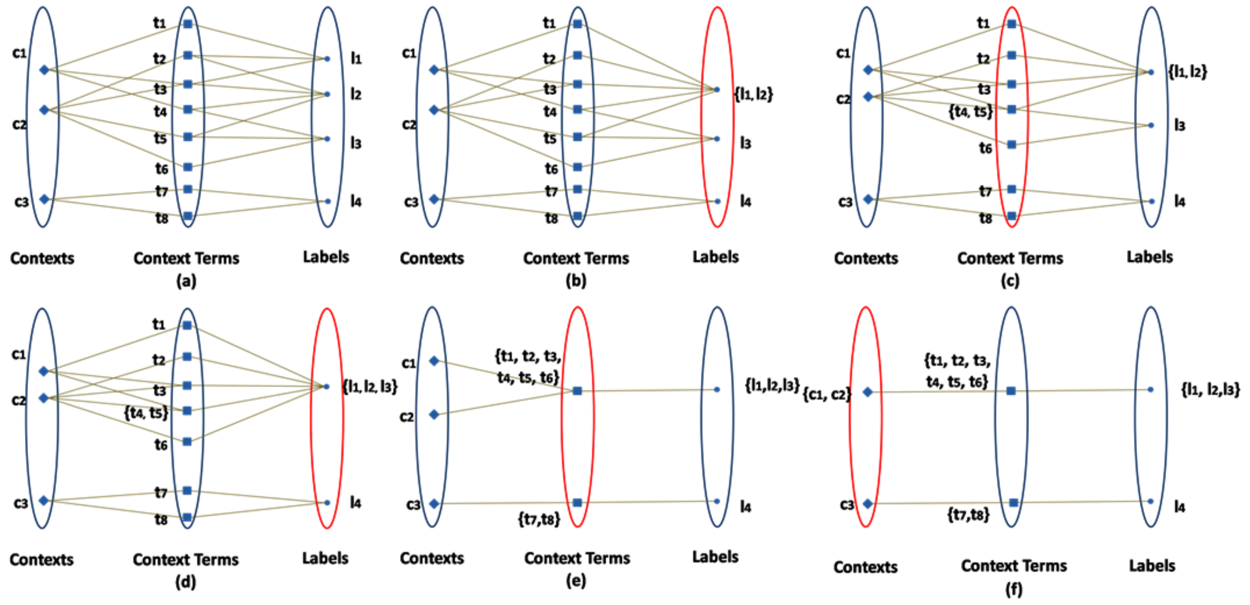


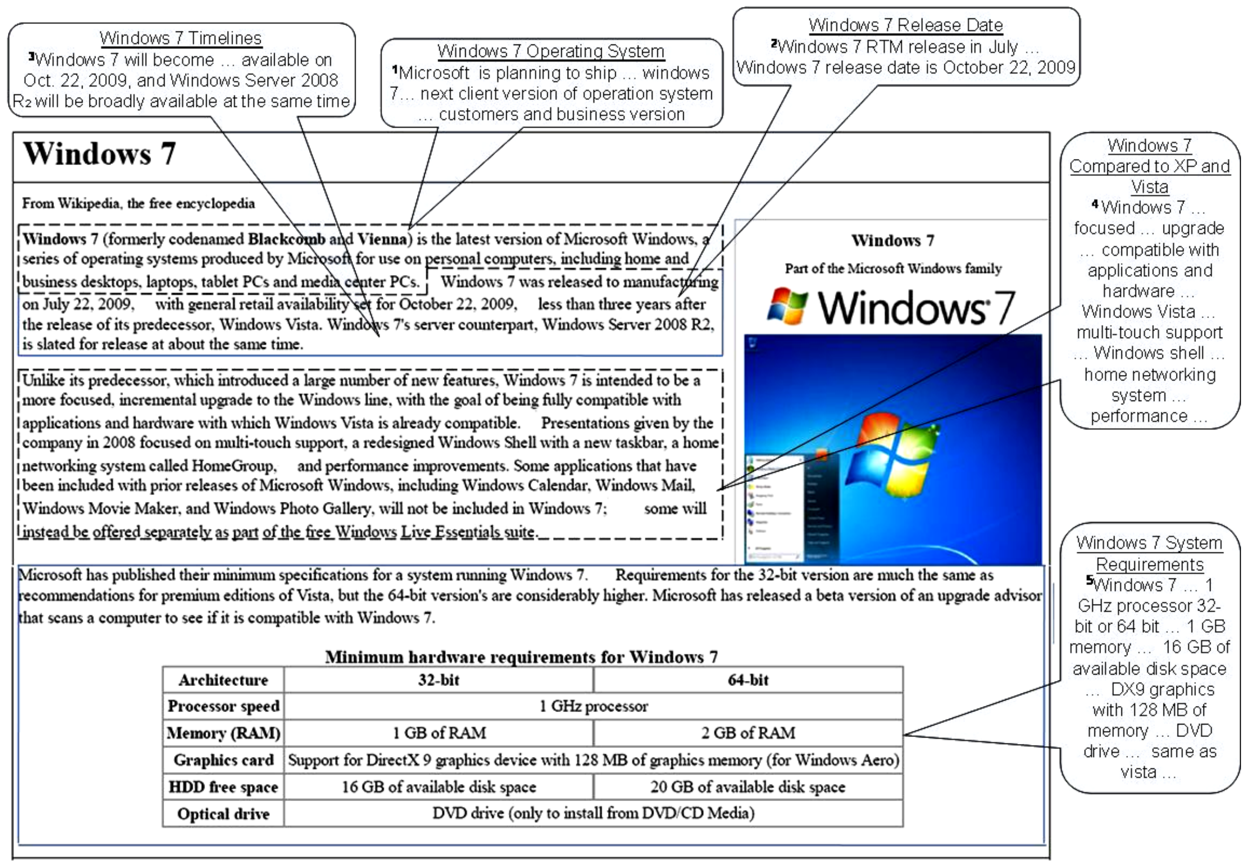
Figure 6.2 A process of constructing label clusters

After label clusters are created, we applied the *TF-IDF Weight* [25] to rank labels in each cluster. Assuming that we have clusters *CL= {cl$_1$, cl$_2$, … , cl$_m$}* and one label *l* from the cluster *cl$_j$*, we propose the following equation to calculate the *l*'s score.

$$S(l) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{num_j(w_i)}{\sum_k num_j(w_k)} \times \log\frac{|CL|}{|\{cl : w_i \in cl\}|}\right) \quad (6.6)$$

In equation 6.6, *n* denotes the number of terms that *l* contains; *num$_j$(w$_i$)* denotes the number of occurrence of term *w$_i$* in the cluster *cl$_j$*; *∑$_k$num$_j$ (w$_i$)* the sum of number of occurrence of all terms in *cl$_j$*; *|CL|* the number of clusters; and *|{cl: w$_i$ ϵ cl}|* the number of clusters containing term *w$_i$*.

Given a query "Windows 7", Figure 6.3 shows an example of retrieving relevant queries as the query-related content labels for the Web page "Windows 7-Wikipedia." First, we identify the locations of the query and query-related context terms in the Web page. Also, we determine the contexts containing query terms based on the methods proposed in section 6.1.1. Second, based on the equation (3), we browse the search history and retrieve past queries related to the given one as candidates of query-related topic labels, such as "Windows 7 Operating System", "Windows 7 development." Third, for the candidate labels, we collect their subsequently clicked Web pages based on the clickthrough data, and applied the methods proposed in section 6.1.1 to retrieve context terms around them in those pages. Based on the similarities of context terms for labels and the given query “Windows 7”, we apply the tripartite graph clustering algorithm to construct clusters of labels so that label clusters refer to query-related topics in the target page. Finally, we obtain four contexts, which are marked by the rectangles in Figure 6.3, and four clusters of labels refer to them. The labels and the terms occurring in both the target page and their subsequently clicked Web pages are contained by rectangular callouts. For example, the contents of second and the third sentences in the target page are referred by two labels, “Windows release date” and “Windows timelines” as shown in Figure 6.3.


Windows 7 Timelines
³Windows 7 will become … available on Oct. 22, 2009, and Windows Server 2008 R2 will be broadly available at the same time
Windows 7 Operating System
¹Microsoft is planning to ship … windows 7… next client version of operation system … customers and business version
Windows 7 Release Date
²Windows 7 RTM release in July …
Windows 7 release date is October 22, 2009
Windows 7 Compared to XP and Vista
⁴Windows 7 … focused … upgrade … compatible with applications and hardware … Windows Vista … multi-touch support … Windows shell … home networking system … performance …
Windows 7 System Requirements
⁵Windows 7 … 1 GHz processor 32-bit or 64 bit … 1 GB memory … 16 GB of available disk space … DX9 graphics with 128 MB of memory … DVD drive … same as vista …
Windows 7
Part of the Microsoft Windows family
Windows 7

Windows 7

From Wikipedia, the free encyclopedia

Windows 7 (formerly codenamed **Blackcomb** and **Vienna**) is the latest version of Microsoft Windows, a series of operating systems produced by Microsoft for use on personal computers, including home and business desktops, laptops, tablet PCs and media center PCs. Windows 7 was released to manufacturing on July 22, 2009, with general retail availability set for October 22, 2009, less than three years after the release of its predecessor, Windows Vista. Windows 7's server counterpart, Windows Server 2008 R2, is slated for release at about the same time.

Unlike its predecessor, which introduced a large number of new features, Windows 7 is intended to be a more focused, incremental upgrade to the Windows line, with the goal of being fully compatible with applications and hardware with which Windows Vista is already compatible. Presentations given by the company in 2008 focused on multi-touch support, a redesigned Windows Shell with a new taskbar, a home networking system called HomeGroup, and performance improvements. Some applications that have been included with prior releases of Microsoft Windows, including Windows Calendar, Windows Mail, Windows Movie Maker, and Windows Photo Gallery, will not be included in Windows 7; some will instead be offered separately as part of the free Windows Live Essentials suite.

Microsoft has published their minimum specifications for a system running Windows 7. Requirements for the 32-bit version are much the same as recommendations for premium editions of Vista, but the 64-bit version's are considerably higher. Microsoft has released a beta version of an upgrade advisor that scans a computer to see if it is compatible with Windows 7.

Minimum hardware requirements for Windows 7

Architecture	32-bit	64-bit
Processor speed	1 GHz processor	
Memory (RAM)	1 GB of RAM	2 GB of RAM
Graphics card	Support for DirectX 9 graphics device with 128 MB of graphics memory (for Windows Aero)	
HDD free space	16 GB of available disk space	20 GB of available disk space
Optical drive	DVD drive (only to install from DVD/CD Media)	

Figure 6.3 An example of retrieving relevant queries as labels of the query “Windows 7” related content in Web page “Windows 7”

6.1.3 Constructing summaries based on labels

In section 6.1.2, we constructed label clusters based on the query-related topics that they refer to. In this section, we construct the query-specific summaries based on top rank labels in the clusters.

Since multiple query-related topics may exist in the target page, we construct the summaries for one topic just based on its corresponding label clusters. Based on the algorithm one, we obtain label clusters and their corresponding contexts. Since the contexts containing contents that the clusters referring to, we attempt to locate sentences containing top rank label terms from the contexts and select appropriate sentences as summaries. In the following example, we identify the terms for the label "Windows 7 System Requirement" in the last paragraph of the target page "Windows 7 – Wikipedia."

Label Terms: *Windows 7, System, Requirement*

Text: *"Microsoft has publishing their minimum specifications for a system running Windows 7. Requirements for the 32-bit version are much the same as recommendations for premium editions of Vista, but the 64-bit version are considerably higher. Microsoft has released a beta version of an upgrade advisor that scans a computer to see if it is compatible with Windows 7. Minimum hardware requirements for Windows 7."*

Then, we applied the Density-based Selection (DBS) approach [30] to rank and select the topic-related sentences inside the contexts. In DBS approach, sentences in the contexts are assigned significance scores and the sentences with high significance scores are collected to construct the summary. Given a set of terms in labels, a sentence is scored based on the following equation.

$$Score(s_i)_m = \frac{1}{K \times (K+1)} \times \sum_{j=1}^{K-1} \frac{Score(w_j)_m \times Score(w_{j+1})_m}{distance(w_j, w_{j+1})^2} \qquad (6.7),$$

where *Score*$(s_i)_m$ is the significance score of the sentence s_i for the query-related topic *m*; *K* is the total number of words in the sentence $s_{i;}$ w_j is a term in the sentence s_j and contained by top rank labels in cluster *m*; *Score*$(w_j)_m$ is the term frequency of w_j occurring in the label cluster *m*; *distance*(w_j, w_{j+1}) is the number of non-label terms between w_j and w_{j+1}. Based on equation 6.7, we may conclude the significance factors of sentences are determined by the significance scores of label terms in sentences and the number of non-label terms between them.

Based on our initial experiments, if the number of label terms occurring in the target Web page divided by the total number of label terms is smaller than 0.3, we consider the page does not contain the labels since the authors of that page may not use the same terms to present the Web page. For example, in Figure 6.3, we retrieve the query "Windows 7 compared to XP and vista" to referring to the contents in the second paragraph, but we cannot locate the label in the second paragraph of the source Web page. One possible solution is retrieving sentences containing label terms from labels' subsequently Web pages for summaries. However, using sentences from other Web pages to summarize the target page may deviate from its original meanings. Thus, we consider context terms co-occurring in the target page and in labels' pages as significant terms. Then, we apply the DBS approach to select appropriate sentences containing the significant terms as summaries. In the following example, we identify the significant terms for the label "Windows 7 Compared to XP and Vista" in the second paragraph of "Windows 7 – Wikipedia."

Significant Terms: *Windows 7, Windows Vista, Multi-touch Support, Windows Shell, Home Networking System, Performance*

Text: *"Presentations given by the company in 2008 focused on multi-touch support, a redesigned Windows Shell with a new taskbar, a home networking system called HomeGroup, and performance improvements."*

6.2 Experiments

The data collections and experiments are presented in this section. First, we constructed two test data sets based on searching logs of a commercial search engine and CNN news. Then, we implemented our approach and compared ours performance with other two well-known approaches.

6.2.1 Data collections

We constructed the first test data set based on the log data of a commercial search engine used in chapter 4. For this collection, first, we filtered queries not satisfying the following three requirements:

1. The queries were written in English and contained alphabet characters;
2. The queries whose lengths were less than eight words;
3. The queries that resulted in at least two unique clicks per session.

Since this paper only studies the English-based query-specific Web page summarization, the research data should satisfy the first criterion. Also, the research data should satisfy the second criterion because it is difficult for search engines to identify key concepts of long queries and speculate users' search needs. Moreover, since not all queries in the log data are effective for searching desired Web pages, enough clicked search results can verify the effectiveness of search queries. Then, we selected two hundred popular queries whose numbers of occurrence are larger than 5 and satisfying above criterions as our test data. Based on the ODP categories of these queries' subsequently clicked Web pages and human judgements, we roughly divided the queries into following categories: News 45 queries, Society 20 queries, Computer 74 queries, Travel 32 queries, and Sports 29 queries.

We submitted the selected queries to GOOGLE. For each submitted query, we collected three search result pages based on their rankings, one from top ten, one from eleven to twenty, and one from twenty-one to thirty. Then, for the selected six hundred search result pages, eighteen human evaluators were employed to construct query-specific summaries. Those evaluators were divided into six groups, and each group was responsible for summarize one hundred search result pages. First, each evaluator was requested to identify query-related contents of the pages. Second, based on the relations of identified contents, evaluators classified contents and thus recognized the query-related topics of the pages. For obtain consistency over three evaluators in one group, the topics concluded by two or more evaluators would be considered as benchmarks for evaluating experiment results. Third, although no constraint on the number of selected sentences was set up, evaluators were asked to only extract key sentences that they considered to be necessary for representing query-related topics. Also, the sentences selected by two or more evaluators for one query-related topic would be considered as benchmarks of summaries. Thus, we constructed the test data set *DAT1*, which contained two hundred queries, six hundred Web pages and summaries created by evaluators. Table 6.1 shows the number of query-related topics and the average number of sentences for one topic identified by evaluators.

Table 6.1 The number of query-related topics and the average number of sentences for one topic

	Number of Topics	Average Number of Sentences
Group 1	152	1.7
Group 2	137	2.4
Group 3	149	2.2
Group 4	186	2.9
Group 5	112	1.4
Group 6	193	3.1

Also, we constructed a larger data set *DAT2* based on the news from CNN.com to evaluate our query-specific summarization approach. Since CNN news have story highlights to summarize the whole news, we may extract highlights related to users' queries as query-specific summaries. First, we extracted queries that lead to click CNN.com news based on the search engine log data. Second, we applied the *Term Frequency Weight* [25] to calculate the similarities of selected queries and highlights of

CNN news. Then, based on the similarity scores, we selected top ranking 1000 queries and their corresponding 1000 CNN news as our test data *DAT2*. The highlights that are highly relevant to queries are considered as query-specific summaries for the news.

6.2.2 Comparative approaches and evaluation metrics

Comparative Approaches. We implemented two effective and widely used query-specific summarization approaches for evaluating the performance of our label-clustering based summarization approach. The first approach, Query Relevance based summarization (QR), ranks sentences based on their relevance to the query and constructs summaries by selecting top rank sentences. The second approach, Graph-based summarization (GS) [51], constructs document-weight-graph based on sentence semantic relations, and then applies graph algorithms to find a minimal spanning tree *T* in the document graph so that every query term exists in the tree *T* and no node can be removed from the tree.

Evaluation Metrics for Query-Related Topics Identification. The purpose of these metrics is verifying that our approach may retrieve relevant search queries as labels of query-related topics and construct label clusters for identifying the topics in the target page. Based on the algorithm one, we collected contexts created by the label clustering algorithm and compared them with the contexts for query-related topics that evaluators found. Strong similarities between them may indicate our approach correctly identify query-related topics in the target page.

For contexts $CI = \{ci_1, ci_2,\ldots,ci_m\}$ created by the label clustering algorithm and contexts $CE = \{ce_1, ce_2, \ldots, ce_n\}$ identified by evaluators, we consider *CE* as benchmarks of contexts and applied the F-measure [1] to verify the correctness of *CI*, which is given by the equation (6.8),

$$F(ci_i, ce_j) = \frac{2 \times Precision(ci_i, ce_j) \times Recall(ci_i, ce_j)}{Precision(ci_i, ce_j) + Recall(ci_i, ce_j)} \quad (6.8),$$

where *Precision(ci_i, ce_j)* is the number of common sentence in cluster ci_i and ce_j divided by the number of sentences in cluster ci_i, and *Recall(ci_i, ce_j)* is number of common sentence in cluster ci_i and ce_j divided by the number of sentences in ce_j. The correctness of query-related topics identification is determined by the following equation,

$$C(CI) = \frac{1}{n}\sum_i \max\{F(ci_i, CE)\} \qquad (6.9),$$

where *max{F(ci_i,CE)}* is the max F-measure value of ci_i with one context in *CE*.

Web Page Summarization Evaluation Metrics. We applied precision, Recall, and F-measure to evaluate summarization performance. The summaries created by evaluators would be considered as reference used for evaluating our summaries' quality. Based on the number of common sentences occurring in our summaries and the reference summaries, the Precisions, Recalls and F-measures of the test set *DAT1* can be calculated. However, highlights from news were created by editors but not extracted from the target Web pages, so we cannot use precision, Recall, and F-measure to evaluate the summaries' quality on *DAT2*.

We used *ROUGE*, a well known tool adopt by *DUC* [32], for evaluating summaries' quality on both *DAT1* and *DAT2*. It measures summaries' quality by calculating the similarities between the candidate summaries and the reference summaries. Rouge-N is one n-gram recall measure defined in the following equation,

$$ROUGE-N = \frac{\sum_{S\in\{ref\}}\sum_{gram_n\in S} Count_{match}(gram_n)}{\sum_{S\in\{ref\}}\sum_{gram_n\in S} Count(gram_n)} \qquad (6.10),$$

where n represents n-gram length, *$Count_{match}(gram_n)$* is the number of n-grams occurring in both candidate and reference summaries, and *$Count(gram_n)$* is the number of n-grams in candidate summaries.

6.2.3 Label Extraction Evaluation

In the dataset *DAT1*, we selected 200 queries from search logs and obtained 600 their search result pages. Then, we divided search result pages into three groups, each of which contained 200 results. The first group contained search results whose ranks were in top ten, the second from eleven to twenty and the third from twenty-one to thirty. Next, we retrieved query-related context terms from the selected search results based on the methods we proposed in the section 6.1.1. Based on the equation (6.3), we calculated relevance of past queries with the given queries and their context terms, and extracted top 10% relevant queries to refer to query-related contents in the search result pages. The following Figure 6.4 shows the number of retrieved relevant queries for three groups of search results.

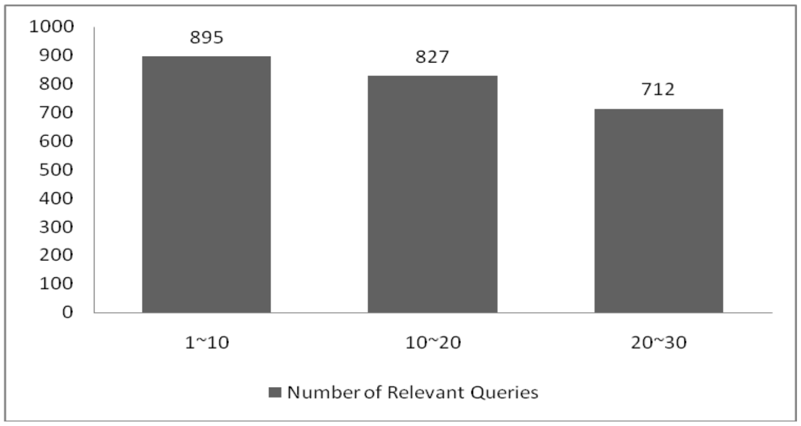


Figure 6.4 The number of relevant queries for three groups of search results

As shown in Figure 6.4, we retrieved more relevant queries for search result pages whose ranks were in top ten than pages whose ranks were from eleven to twenty and from twenty-one to thirty. This result is consistent with our hypothesis one "The query-related topics from its top ranking Web search results may be described by other relevant queries." Also, it indicates that more relevant queries can be collected for pages with higher ranks. Since this paper attempts to construct query-specific summaries for search result pages whose ranks in top 30, our approach may retrieve enough relevant queries to uncover query-related topics in the result pages.

6.2.4 Evaluation for query-related topics identification

For the dataset *DAT1*, we asked evaluators to determine query-related contents of selected result pages and construct clusters of them so that one content cluster containing one query-related topic. Based on the query-related topics that target pages contained, we divided pages into ten groups and each of them contained sixty pages. The first group owned the least query-related topics; the second one owned second least topics; and the last one owned the most topics.

Also, we applied the methods proposed in section 6.1.1 and 6.1.2 to locate query-related contents and retrieve top 10% relevant queries to refer to them. Then, we applied the algorithm one to construct label clusters and merge the corresponding contents in the target pages. Since our test sets contained two hundred queries and six hundred result pages, and the average number of query-related topics for each page was 2.4, we selected a reasonable value of parameter *δ* for constructing label clusters in Algorithm 1. Based on our initial experiment, we set *δ=0.3* so that we may construct almost same number of label clusters as the number of query-related topics in *DAT1*.

For evaluating the topics identification performance of our algorithm, we implemented a term-similarity based query-related contents clustering approach. First, we still applied the method proposed in section 6.1.1 to identify query-related contents. Second, we calculated query-related contents' similarities based on their terms' similarities and attempted to merge the relevant contents.

We implemented our algorithm one and the above term-similarity based query-related contents clustering approach on *DAT1*. The following Figure 5 shows their performances of query-related topics identification.

As shown in Figure 6.5, the performances of query-related topics identification by the label clustering algorithm are better than by the term-similarity based clustering algorithm on all ten groups of search result pages.

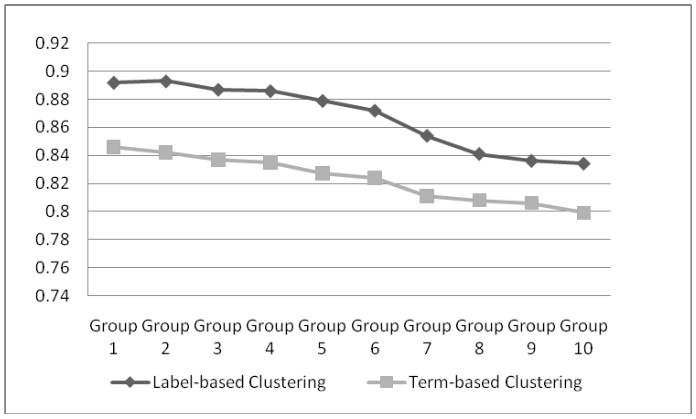


Figure 6.5 F-measure values of query-related topics identification for two clustering approaches

One possible reason is that the term-similarity based clustering algorithm attempts to merge contents based on their terms' similarities. Thus, this approach cannot merge two contents if they do not share a lot of common terms although they may convey relevant meanings. However, the label clustering algorithm attempts to retrieve relevant Web pages (labels' subsequently clicked pages) for the target page and merge its contents if its context terms can co-occur in the these relevant Web pages. In fact, the label clustering algorithm extends contexts of the target page by importing context terms from relevant Web pages. Thus, although two contents may not share a lot of common terms in the target page, after contexts extension, they may be relevant since their extended contexts may contain several common terms.

Figure 6.5 shows that both of the clustering approaches perform worse on the groups containing more query-related topics than the groups containing fewer topics. Also, the performance of label clustering algorithm drops slightly faster than the term-based clustering algorithm. One reason is that the risk of merging two irrelevant contents by the label clustering algorithm is higher than the term-based clustering algorithm since the former approach extends contexts of the target page.

6.2.5 Web Page Summarization Evaluation

Figure 6.6 shows the summaries' qualities for label clustering summarization approach (IS) with other two comparative approaches, *QR* and *GS*, on test sets *DAT1* and *DAT2*. In Figure 6.6, the tag *IS*

(Label found) indicates that retrieved labels can be located in the target Web pages and we constructed summaries by extracting sentences containing label terms. The tag *IS (Label not found)* indicates labels cannot be located in the target pages, so we extracted sentences containing common context terms occurring in both the target page and labels' contexts as summaries.

Compared to the *GS* approach, *IS (Label found)* gains an improvement of 0.08 and 0.03 measured by precision and F-measure respectively on *DAT1*. Since the *IS* attempts to construct summaries based on labels composed by human, it may identify keywords or key sentences of query-related contents correctly and extract important sentences as summaries. However, since it attempts to extract only key sentences as summaries while ignore some sentences that evaluators may consider valuable for complementing summaries, the recall and ROUGE-1 values of *IS (Label found)* are slightly smaller than the *GS*. Although *IS (Label not found)* does not gain improvement as well as *IS (Label found)* in precision and F-measure, it performs slightly better than the latter approach in Recall and ROUGE-1. Since labels cannot found in Web pages, retrieving sentences containing important context terms may construct irrelevant summaries for queries, so precision and F-measure for *IS (Label not found)* may be lower than *IS (Label found)*. However, retrieving sentences containing context terms may also build lengthy summaries and thus more query-related contents may be included, so recall and ROUGE-1 values may be higher. Compared to summaries' quality on *DAT1*, these approaches perform worse on *DAT2*. As shown in the Figure 6.6, the ROUGE-1 value for *QR* drops 0.21 on *DAT2*, *GS* 0.24, *IS (Label found)* 0.18, and *IS (Label not found)* 0.26. One reason is that highlights from news are generic summaries for the whole document but not for specific queries. The other reason is that editors may use their words instead of sentences from the news to create summaries. However, the ROUGE-1 metric cannot judge whether the extractive summaries and news' highlights convey the same meanings but just simply compare their words. Compared to *QR* and *GS* approaches, the *IS (Label found)* approach obtain a relatively better performance on *DAT2*. One possible reason is that *IS (Label found)* may identify key query-related topics of the news and these topics probably occur in the editors' concise highlights.

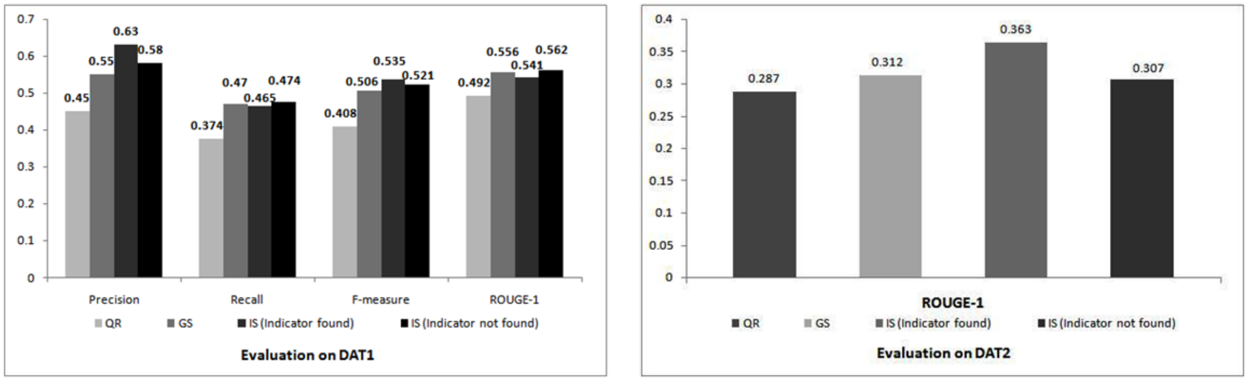


Figure 6.6 Overall summarization performances on DAT1 and DAT2

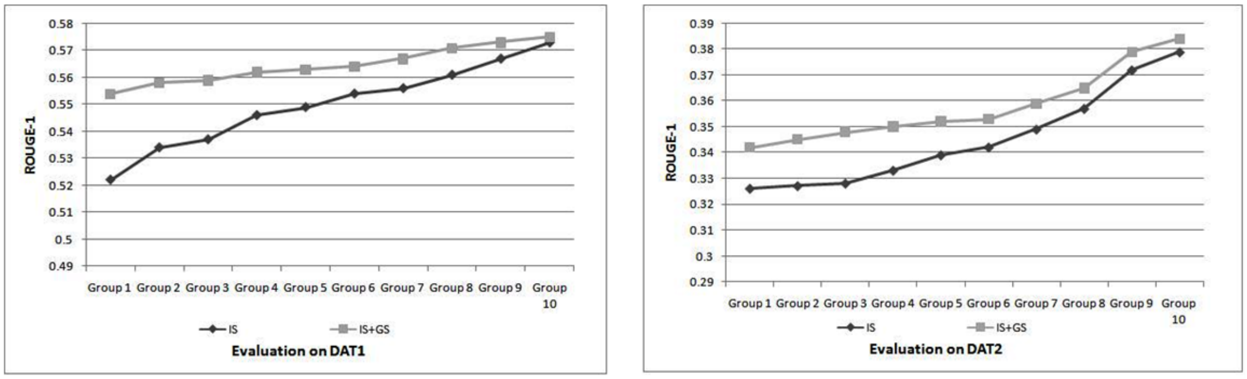


Figure 6.7 ROUGE-1 values for IS and IS+GS summarization on DAT1 and DAT2

In order to evaluate the impacts of the number of the labels for the quality of IS based summaries, we divided the *DAT1* into ten groups and each group contained 20 queries and 60 target Web pages. The first group contained Web pages that had least labels to uncover their query-related contents; the second one contained pages that had second least labels; and the last one contained pages that had most labels. Figure 6.7 shows that the *IS* approach performs worst for the group of pages with least labels on both *DAT1* and *DAT2*. With the number of labels increasing, *IS* performs slightly better. Compared to the first group, *IS* achieves 0.05 and 0.053 improvements for the last group on *DAT1* and *DAT2* respectively. These results indicate that more labels referring to query-related contents of target pages may result in high quality summaries by the *IS* approach.

For studying the potential of improving the *IS* approach's performance, we combined the *IS* and *GS* approaches on the ten groups of Web pages and compared results with *IS*. Since *ROUGE-1* is recall based evaluation metric, combining results from *IS* and *GS* may improve *ROUGE-1* values, so results of *IS+GS* are better than *IS* on all ten groups. However, with more labels found for Web pages, the difference between *IS* and *IS+GS* become smaller, which may indicate *IS* perform well if enough labels found.

6.3 Discussion

For a given query, constructing query-specific summaries for its searched result pages is an important task for most Web search engines. This paper incorporates relevant queries from search history to refer to the current query-related contents in the result pages and constructs the summaries based on the relevant queries. Our experiments show that this approach can identify query-related topics more precisely than the traditional term-similarity based summarization approach. Since we proposes this approach for constructing query-specific summaries on the top thirty search results, the experiment results reveal that we may retrieve enough queries for referring to Web pages and guarantee the summaries' quality. Even if we cannot retrieve enough relevant queries, as our experiments show, the combination of our approach with another traditional query-specific summarization approach will construct good quality summaries.

The contributions of this paper are listed as following:

1. This paper selects relevant queries from search history as labels to identify query-related topics of the target Web page.

2. This paper evaluates relevance of retrieved queries with the given query in the target Web page by comparing context terms around the retrieved queries from their subsequently clicked Web pages with terms around the given query from the target Web page.

Chapter 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

Based on users' clickthrough data, this thesis proposes Query-Concept bipartite graphs to investigate queries' relations. Compared to lexical matching approaches, this new approach explores queries' exact meanings through their subsequently clicked Web pages and then determines their relevance based on their meanings. Thus, this approach can evaluate queries' relations correctly even they do not contain one common word. As discussed in section 3.1, the probability of two queries having same concepts from their subsequently clicked Web pages is larger than that of them resulting in the same Web pages. Therefore, Query-Concept bipartite graphs can find more relevant queries for a given query than Query-URL bipartite graphs.

Query-Concept bipartite graphs are used in three applications of Web search engine improvements: personalized query suggestion, long query Web searches, and query-specific Web page summarization. As discussed in section 4.3, the approach using Query-Concept bipartite graphs obtains more meaningful suggestions for incomplete queries than the Query-URL based approach. As shown in section 5.2, compared with the concept weighting approach for long query reformulation, the Query-Concept based long query substitution approach retrieves more accurate short terms to represent the long query's meanings. This thesis proposes a query-specific Web page summarization approach using relevant short queries to represent the query-related contents in target Web page. Compared to machine learning based summarization approach, the new approach constructs more concise and precise query-specific Web page summaries.

7.2 Future Works

We plan to integrate Personalized Query Suggestion model, Long Query Web Search model, and Query-Specific Web Page Summarization model into current Web search engines. The integrated Web search engine – Smart Search will contain the following modules: Smart Core, Personalized Query

Suggestion, Long Query Reduction, Query-Specific Web Page Summarization, Clickthrough Database, and Meta Web Search Engine. Figure 7.1 shows a sequence diagram to describe the modules and their interactions.

Different from engineers of commercial Web search engines, researchers from academic institutions generally do not own enough users' search data to evaluate their methods. Implementing such a search engine may help us catch users' clickthrough data and evaluate various approaches.

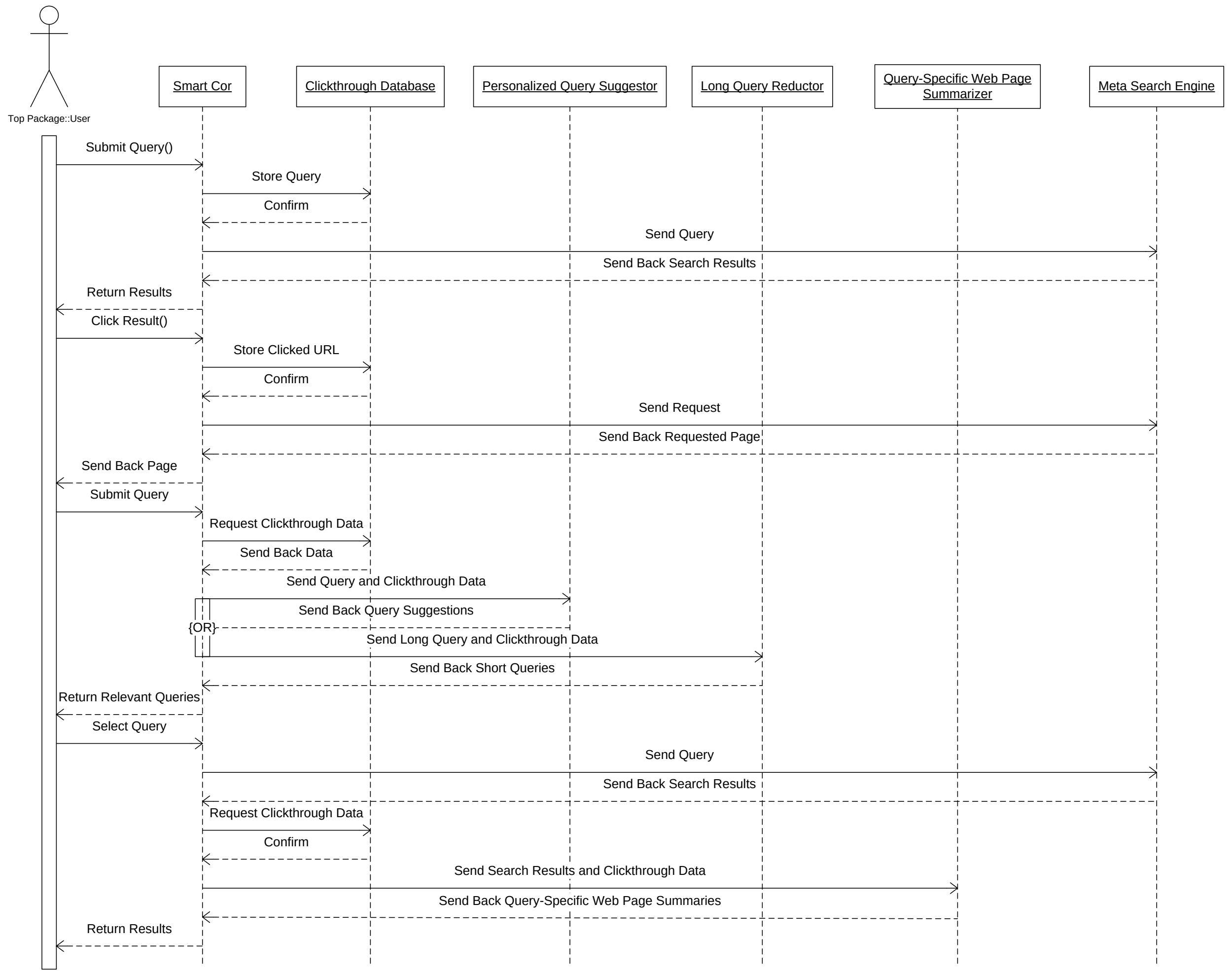


Figure 7.1 The Sequence Diagram of the Smart Search Engine

REFERENCES

[1] R. Baeze-Yates, B. Ribeiro-Neto, “Modern Information Retrieval”, *New York: ACM Press, Addison-Wesley*, 1999.

[2] D. Beeferman and A. Berger, “Agglomerative Clustering of a Search Engine Query Log”, *Proceeding of ACM SIGKDD Conference*, 407-416, 2000.

[3] M. Bendersky and W. B. Croft, “Discovering Key Concepts in Verbose Queries”, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 491-498, 2008.

[4] F. Bonchi, C. Castillo, D. Donato, and A. Gionis, “Topical Query Decomposition”, *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining,* 52- 60, 2008.

[5] O. Boydell and B. Smyth, “From Social Bookmarking to Social Summarization: An Experiment in Community-Based Summary Generation”, In *Proceedings of the 12th international conference on Intelligent user interfaces,* pages 42-51, Honolulu, USA, 2007.

[6] C. Carpineto, R. Mori, G. Romano, and B. Bigi, “An Information-Theoretic Approach to Automatic Query Expansion”, *ACM Transactions on Information Systems (TOIS)*, Vol. 19, No. 1, pp. 1-27, 2001.

[7] G. Chen and B. Choi, “Web Page Genre Classification”, *Proceedings of the ACM symposium on Applied computing*, pp. 2353-2357, 2008.

[8] Y. Chen, G. Xue, and Y. Yu, “Advertising Keyword Suggestion based on Concept Hierarchy”, *Proceedings of the international conference on Web search and web data mining,* pp. 251-260, 2008.

[9] H. Chu, and M. Rosenthal, “Search Engines for the World Wide Web: a Comparative Study and Evaluation Methodology”, *Proceedings of the ASIS 1996 Annual Conference*, 127-35, 1996.

[10] J. Conroy, J. Schlesinger, and D. Leary, “Topics-Focused Multi-document Summarization Using an Approximate Oracle Score”, In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 152-159, Sydney, Australia, 2006.

[11] D. Crabtree, P. Andreae, and X. Gao, “Exploiting Underrepresented Query Aspects for Automatic Query Expansion”, *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 191-200, 2007.

[12] H. Cui, J. Wen, J. Nie, and W. Ma, “Query Expansion by Mining User Logs”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 4, pp. 829-839, 2003.

[13] D. Daniels, “Search Engine Optimization.”Retrieved August 10, 2009, from David Daniels, Personal Branding: http://fairwindsweb.com/searchengineoptimization.html.

[14] G. Furnas, T. Landauer, L. Gomez, and S. Dumais, “The Vocabulary Problem in Human-system Communication”, Communication of ACM, Vol. 30, No. 11, pp. 964-971, 1987.

[15] B. Fonseca, P. Golgher, E. Moura, and N. Ziviani, “Using Association Rules to Discover Search Engines Related Queries”, *Proceedings of the First Conference on Latin American Web Congress*, 66, 2003.

[16] E. Gabrilovich and S. Markovitch, “Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis”, Proceedings of the 20th International Joint Conference on Artificial Intelligence, 1606-1611, 2007.

[17] G. Garrett, “InOrder: An Adaptable Visual Interface for Collaborative Search and Query Reformation”, Retrieved from Dissertation & Thesis @ database, 2006.

[18] J. Guo, G. Xu, H. Li, and X. Cheng, “A Unified and Discriminative Model for Query Refinement”, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 379-386, 2008.

[19] M. Harris, “Searching the World Wide Web”, 2009. Retrieved August 10, 2009, from The OWL at Purdue: http://owl.english.purdue.edu/owl/resource/558/01/.

[20] M. Hearst, “TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages”, *Computational Linguistics*, 23(1): 33-64, 1997.

[21] M. Heijden, M. Hinne, W. Kraaij, S. Verberne, and T. Weide, “Using Query Logs and Click Data to Create Improved Document Descriptions”, *In Proceedings of the 2009 workshop on Web Search Click Data*, pages 64-67, Barcelona, Spain, 2009.

[22] M. Hu, A. Sun and E. Lim, “Comments-oriented Blog Summarization by Sentence Extraction”, *In Proceedings of the 16th CIKM,* pages 901-904, Lisbon, Portugal, 2007.

[23] M. Hu, A. Sun and E. Lim, “Comments-Oriented Document Summarization: Understanding Documents with Readers’ Feedback”, In *Proceeding of the 31st SIGIR conference on research and development in information retrieval*, pages 291-298, Singapore, Singapore, 2008.

[24] 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.

[25] K. Jones, "A Statistical Interpretation of Term Specificity and its Application in Retrieval", *Journal of Documentation*, 28(1): 11-21, 1972.

[26] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating Query Substitutions", *Proceedings of the 15th international conference on World Wide Web,* 387-396, 2006.

[27] B. Kules, J. Kustanowitz, and B. Shneiderman, "Categorizing Web Search Results into Meaningful and Stable Categories using Fast-feature Techniques", *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pp. 210-219, 2006.

[28] G. Kumaran and J. Allan, "A Case for Shorter Queries, and Helping User Create Them", *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 220—227, 2006.

[29] G. Kumaran and J. Allan, "Effective and Efficient User Interaction for Long Queries", *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 11-18, 2008.

[30] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim, "Siteq: Engineering High Performance QA system using Lexico-semantic Pattern Matching and Shallow NLP", *Proceedings of TREC'01*, 437–446, 2001.

[31] C. Lin and E. Hovy, "The Automated Acquisition of Topic Signature for Text Summarization", *In Proceeding of the 18th conference on Computational linguists*, pages 495-501, Saarbrücken, Germany, 2000.

[32] C. Lin and E. Hovy, "Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics", In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71-78, Edmonton, Canada, 2003.

[33] D. Metzler, S. Dumais and C. Meek, "Similarity Measures for Short Segments of Text", Proceedings of 29th European Conference on IR Research, 16-27, 2007.

[34] G. Miller, "WordNet Nouns: Classes and Instances", *Computational Linguistics*, Vol. 32, No. 1, pp. 1-3, 2006.

[35] J. Otterbacher, G. Erkan, and D. Radev, "Biased LexRank: Passage Retrieval using Random Walks with Question-Based Priors", *Information Processing and Management*, 45(1): 42-54, 2009.

[36] E. Park, S. Moon, D. Ra, and M. Jang, "Web Document Retrieval Using Sentence-query Similarity", *Proceedings of the 11th Text REtrieval Conference (TREC-11)*, 2002.

[37] B. Ricardo, H. Carlos, and M. Marcelo, “Query Recommendation using Query Logs in Search Engines”, *EDBT Workshops,* pp. 588-596, 2004.

[38] T. Roelleke, and J. Wang, “TF-IDF Uncovered: A Study of Theories and Probabilities”, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 435-442, 2008.

[39] S. Ross, *A First Course in Probability*, Macmillan, 195–240, 1976.

[40] M. Sahami and T. Heilman, “A Web-based Kernel Function for Measuring the Similarity of Short Text Snippets”, *Proceedings of the 15th International World Wide Web Conference*, 377-386, 2006.

[41] F. Scholer and H. Williams, “Query Associations for Effective Retrieval”, *In Proceedings of the 11th CIKM*, pages 324-331, McLean, USA, 2002.

[42] “Search Engine User Behaviour Study”, 2006. Retrieved August 10, 2009, from IProspect Search Engine Marketing Firm: http://www.iprospect.com/premiumPDFs/WhitePaper_2006_SearchEngineUserBehavior.pdf

[43] J. Shapiro, and I. Taska, “Constructing Web Search Queries from the User's Information Need Expressed in a Natural Language”, *Proceedings of the 2003 ACM symposium on Applied computing*, 1157-1162, 2003.

[44] J. Sun, D. Shen, H. Zeng, Q. Yang, Y. Lu, and Z. Cheng, “Web-Page Summarization Using Clickthrough Data”, In *Proceeding of the 28st SIGIR conference on research and development in information retrieval*, pages 194-201, Salvador, Brazil, 2005.

[45] W. Soon, H. NG, and D. Lim, “A Machine Learning Approach to Coreference Resolution of Noun Phrases”, *Computational linguistics Special issue on computational anaphora resolution*, 27(4): 521-544, 2001.

[46] A. Spink and H. C. Ozmultu, “Characteristics of Question Format Web Queries: An Exploratory Study”, *Information Processing and Management*, 38(4): 453-471, 2002.

[47] K. Svore, L. Vanderwende, and C. Burges, “Using signals of Human Interest to Enhance Single-document Summarization”, *Proceedings of AAAI 2008,* pages 1577-1580, Chicago, USA, 2008.

[48] B. Tan and F. Peng, “Unsupervised Query Segmentation Using Generative Language Models and Wikipedia”, *Proceeding of the 17th international conference on World Wide Web,* 347-356, 2008.

[49] P. Tan, M. Steinbach, and V. Kumar, “*Introduction to Data Mining: Concepts and Techniques*”, Addison Wesley, pp. 547-579, 2006.

[50] A. Tombros and M. Sanderson, “Advantages of Query Biased Summaries in Information Retrieval”, In *Proceeding of the 21st SIGIR conference on research and development in information retrieval*, pages 2-10, Melbourne, Australia, 1998.

[51] R. Varadarajan and V. Hristidis, “A System for Query-Specific Document Summarization”, *In Proceedings of the 15th CIKM,* pages 622-631, Arlington, USA, 2006.

[52] X. Wang, and C. Zhai, “Learn from Web Search Logs to Organize Search Results”, *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 87-94, 2007.

[53] X. Wang, J. Yang and J. Xiao, “CollabSum: Exploiting Multiple Document Clustering for Collaborative Single Document Summarizations”, In *Proceeding of the 30th SIGIR conference on research and development in information retrieval,* pages 143-150, Amsterdam, The Netherlands, 2007.

[54] F. Wei, W. Li, Q. Lu, and Y. He, “Query-Sensitive Mutual Reinforcement Chain and Its Application in Query-Oriented Multi-Document Summarization”, In *Proceeding of the 31st SIGIR conference on research and development in information retrieval*, pages 283-290, Singapore, Singapore, 2008.

[55] J. Wen, J. Nie, and H. Zhang, “Query Clustering Using User Logs”, *ACM Transactions on Information Systems (TOIS)*, 20(1): 59-81, 2002.

[56] C. Whitelaw, A. Kehlenbeck, N. Petrovic, L. Ungar, “Web-Scale Named Entity Recognition”, *Proceeding of the 17th ACM conference on Information and knowledge management,* 123-132, 2008.

[57] R. White, J. Jose and I. Ruthven, “Query-biased Web Page Summarisation: A Task-Oriented Evaluation”, In *Proceeding of the 24st SIGIR conference on research and development in information retrieval*, pages 412-413, New Orleans, USA, 2001.

[58] G. Xue, D. Xing, Q. Yang, and Y. Yu, “Deep Classification in Large-scale Text Hierarchies”, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 619-626, 2008.

[59] J. Yang, Y. Jiang, A. Hauptmann, and G. Ngo, “Evaluating Bag-of-visual-words Representations in Scene Classification”, *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pp. 197-206, 2007.

[60] I. Zukerman, B. Raskutti, and Y. Wen, “Query Expansion and Query Reduction in Document Retrieval”, *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, 552-560, 2003.

[61] Y. Chen and Y.-Q. Zhang, "A Personalized Query Suggestion Agent based on Query Concept Bipartite Graphs and Concept Relation Trees", *the Special Issue on Intelligent Techniques for Personalization and Recommendation, the International Journal of Advanced Intelligence Paradigms*, 2009.

[62] Y. Chen and Y.-Q. Zhang, "A Query Substitution-Search Result Refinement Approach for Long Query Web Searches", *Proc. of IEEE/WIC/ACM-WI2009*, Milan, Sept. 15-18, 2009.