THE UNIVERSITY OF TOKYO

ENHANCING WEB SEARCH BY PERSONALIZED RE-RANKING AND RELATED KEYWORD SUGGESTION

by

LIN LI

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Graduate School of Information Science and Technology
Department of Information & Communication Engineering

March 2009

Never leave that until tomorrow, which you can do today.

Benjamin Franklin, American president

THE UNIVERSITY OF TOKYO

Abstract

Graduate School of Information Science and Technology
Department of Information & Communication Engineering

Doctor of Philosophy

by LIN LI

Web search engines today provide simple and yet friendly user interfaces which allow users to pose Web search queries simply defined by a list of keywords. Given a search query, search engines primarily rely on the matching of the keywords to terms in Web pages to determine which pages will be returned. The main limitation with keyword-based search is two folds. First, due to the ambiguity of user needs, some keywords have different meanings in different context, such as mouse trap, Jaguar, Java and so on. Present search engines generally handle search queries without considering user preferences or contexts in which users submit their queries. Furthermore, users often fail to choose proper keywords that best express their information needs. Ambiguous keywords used in Web queries, the diverse needs of users, and the limited ability of users to precisely express what they want have been widely recognized as a challenging obstacle in improving search quality.

In this thesis, we study two strategies which are widely used to enhance the retrieval quality of Web search. One is personalized re-ranking. To satisfy the diverse needs of users, we improve the global notion of importance in ranking search results by creating personalized view of importance. A set of efficient mechanisms are devised to re-rank search results by personalized ranking criteria which are typically derived from the modeling of users' search behavior and interests. The other is keyword suggestion. Our work focuses on finding semantically related search queries (though probably dissimilar in their terms). We propose using additional sources to enrich short Web queries and compare different feature spaces to better represent the meanings of a query than the keywords in defining it. Thus, the relatedness between two queries can be estimated based on their enriched representations. The suggested keywords can assist Web users in rephrasing their query formulation. We show that personalized re-ranking and related keyword suggestion are effective strategies to enhance Web search.

Acknowledgements

I could never have completed this work without the support and assistance of many people. First and foremost, I would like to express deepest gratitude to my advisor, Prof. Masaru Kitsuregawa, for his excellent guidance, valuable suggestions, and kind encouragement in academic. With his help, I learned how to define research problems and formalize them; how to design solutions and improve them; and how to write papers and present them. Prof. Masaru Kitsuregawa is a good advisor and a helpful friend a student could hope for. His comprehensive philosophy of the world and human life teaches me how to face and overcome difficulties and setbacks, and helps me grow as a researcher with a positive and optimistic view of life. For his consistent and generous support, I thank him from the bottom of my heart.

I also would like to express grateful thank to Assoc. Prof. Masashi Toyoda, Assoc. Prof. Miyuki Nakano, and Dr. Shingo Otsuka for their kind advices, helpful assistance, technical support and countless hours of useful discussion on the direction and many issues regarding to my research. I would specially like to thank Prof. Ling Liu for sparing time to provide constructive comments on my research. The discussions with her brought me fresh and exciting ideas. Also, sincere thanks to my Ph.D. committee professors, Prof. Jun Adachi, Prof. Takashi Chikayama, and Prof. Mitsuru Ishizuka.

This thesis would not have been possible without generous financial support from Japanese Government (Monbukagakusho) Scholarship program. Through its very kind staffs, Monbukagakusho also provides student accommodation and other convenient life supports. These supports are gratefully acknowledged.

The rigor and hard work of my Ph.D. has been balanced by the fun and joy of being in company of Dr. Wenyu Qu, Dr. Zhenglu Yang, Dr. Kulwadee Somboonviwat, Yanhui Gu, Yongkun Wang, to name a select few. I also want to thank all colleagues in Kitsuregawa, Toyoda Laboratory who have offered me friendship, insightful comments, ideas, and helpful discussions during the course of my study.

Finally, all this would have not been worthwhile, but for my family. It is impossible to put into words my feelings of love and gratitude for my parents and my husband. It is their understanding, perpetual support, and unconditional love that make me overcome all the difficulties through all the years of my study in Japan.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The World Wide Web has become a new communication medium with informational, cultural, social and evidential values after a few decades since its inception. Search engines are widely used for Web information access and they are making more information easily accessible than ever before. Although the general Web search today is still performed and delivered predominantly through search algorithms, e.g., Google's PageRank [70], the interests in helping Web users effectively get their desired Web pages have been growing over the recent years.

1.1 Motivation

Keyword based query is a much more popular way to let Web users easily specify their information needs than SQL queries in commercial search engines like Google and Yahoo!. The simple and yet friendly Web user interfaces provided by those search engines allow users to pose search queries simply in terms of keywords. However, the difficulty in finding only those which satisfy an individual's information goal increases. This is because search engines primarily rely on the matching of the keywords to the terms in the desired documents to determine which Web pages will be returned given a search query. The main limitation with keyword-based search queries is two folds. First, due to the ambiguity of user needs, some keywords have different meanings in different context, such as mouse trap, Jaguar, Java and so on. Present search engines generally handle search queries without considering user preferences or contexts in which users submit their queries. Furthermore, users often fail to choose proper terms that best express their information needs. Ambiguous keywords used in Web queries, the diverse needs of users, and the limited ability of users to precisely express what they want to search in a

few keywords have been widely recognized as a challenging obstacle in improving search quality.

Currently, encoding human search experiences and personalizing the search result delivery through ranking optimization is a popular approach in recent data engineering field to enhancing the result quality of Web search and user experience with the Web today. A general process of search result re-ranking is to devise efficient mechanisms to re-order search results by personalized ranking criteria. Such criteria are typically derived from the modeling of users' search behavior and interests. Even though short-term interests based personalization using the most recent search histories may be effective at times [60, 82, 83], it is generally unstable and fails to capture the changing behavior of the users. Furthermore, most of existing long-term interests based personalization using the entire recent and previous search histories fails to distinguish the relevant search history from irrelevant search history [16, 72, 94], making it harder to be an effective measure alone for search personalization.

On the other hand, commercial search engines give related keyword suggestion based on the search queries input by users, thus assisting users in rephrasing their query formulation to improve search quality. *Related search terms* in Google is based on the assumption that sometimes the best search terms for what a user is looking for are related to the ones the user actually entered. In the search box of Yahoo!, *Search Assist* compares an input query to searches all other Yahoo! users have composed and offers suggestions in real time. The purpose of these methods is to help users specify alternative related queries in their search process in order either to clarify their information needs or to rephrase their query formulation to retrieve more related search results. The suggestion services supplied by those popular search engines highlight the importance of related keyword suggestion. Although the techniques used in these proprietary commercial search engines are usually confidential, academic researchers have showed growing interests in query suggestion [2, 3, 5, 29, 33, 52, 55, 61, 63, 75, 80, 97, 101, 105] . A study of the log of a popular search engine [39] reported that most search queries are about two terms per query. Therefore, the difficulty is that since Web users typically submit very short queries to search engines, the very small term overlap between queries cannot accurately estimate their relatedness. Given this problem, the technique to find semantically related queries (though probably dissimilar in their terms) is becoming an increasingly important research topic that attracts considerable attention.

In this thesis, we study two strategies, i.e., personalized re-ranking of search results and related keyword suggestion. The goal of both two strategies is to improve search engines, especially retrieval quality. When working on the former strategy we discuss how metadata widely used in the semantic Web can be exploited to achieve high quality

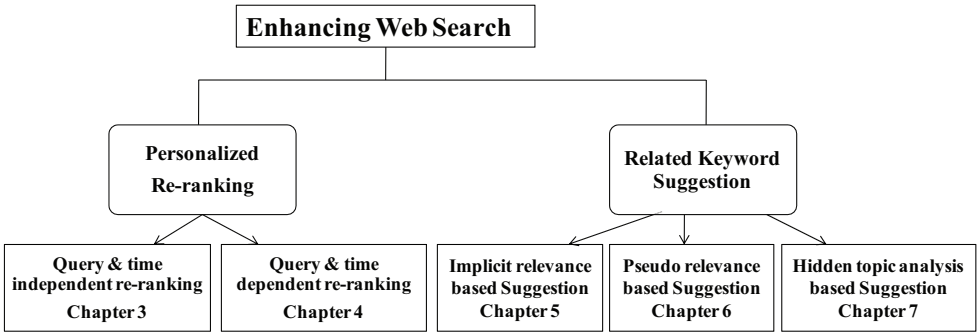

FIGURE 1.1: Thesis framework

personalized Web search. And for the latter strategy we focus on finding different available features in query similarity computation instead of using keywords.

1.2 Thesis Outline

Figure 1.1 shows the framework of this thesis. The rest of this thesis is organized as follows.

Chapter 2 introduces some related work, including personalized search, learning users' interests by building ontology-based user profiles, Web query clustering, Web query expansion, rank aggregation and so on.

Chapter 3 and 4 study personalized re-ranking of search results for Web search enhancement. We present four re-ranking strategies by augmenting user search history.

In Chapter 3 we focus on learning user preferences and building user profiles to re-rank search results. Because user preferences change over time, it becomes important to keep the user profiles up-to-date, and for a search engine to adapt accordingly. In addition, a user profile covers both short-term and long-term user preferences. Using one model to represent two differently featured parts of the user profile will be far from perfect. To address the two problems, we design a ontology based user profile (i.e., the hierarchy of the ODP structure) for the long-term model and a recently visited topic-history buffer for the short-term model. We also propose updating strategy to capture the changes of user preferences. For re-ranking mechanism, we investigate score and rank aggregation methods which combine the general ranking of search engines and the personalized ranking of users, thus custom-tailoring search results.

In Chapter 4 we discuss how metadata like ODP (Open Directory Project) can be further exploited to achieve high quality personalized Web search. The metadata expressing topical categorizations of web pages in ODP is manually entered and edited,

and commonly considered as useful means for automated learning of user interests and user search behaviors. We propose a query context window (QCW) based framework for query-centric *S*elective u*T*ilization of search history in personalized le*A*rning and re-*R*anking (STAR). Our STAR framework consists of three design principles and a suite of algorithms for learning and encoding users' short-term and long-term search interests and re-ranking of search results through a careful combination of recent and previous search histories.

Chapter 5, 6, and 7 describe our approaches to suggesting related keywords. Because Web queries are usually a couple of words, we utilize query enrichment as an effective method to enrich the representation of a Web query by alternative feature spaces, instead of using terms in the query. In the three chapters, we mainly describe the effect of different feature spaces on suggestion precision.

In Chapter 5, we study implicit relevance based query suggestion. Two kinds of feature spaces extracted from the clicked search results of a query are content-sensitive (e.g., nouns) and content-ignorant (e.g., URLs). Our experiment results show that the URL feature space produces lower precision scores than the noun feature space which, however, is not applicable, at least in principle, in settings including: non-text pages like multimedia (image) files, Usenet archives, sites with registration requirement, and dynamic pages returned in response to a submitted query and so forth. It is crucial to improve the quality of the URL (content-ignorant) feature space since it is generally available in all types of Web pages. We are inspired to propose a novel content-ignorant feature space, i.e., Web community which is a collection of Web pages with different URLs, but sharing common interest on a specific topic. Experiment results show that the novel Web community feature space generates much better results than the traditional URL feature space.

In Chapter 6, we study pseudo relevance based query suggestion and improve the performance of the URL feature space from another viewpoint different from that in Chapter 5. We generate query affinity graph based the similarity measure between pairs of queries using query-URL vector model. We propose to utilize the monotonicity of the merging distances of a hierarchical agglomerative clustering (HAC) which can capture the propagation of similarity (distance) implicitly. By using HAC based algorithms we can globally capture the diffusive transition of similarity on each connected component extracted from query affinity graph to rank queries. Furthermore, instead of fixing the degree of similarity of queries based on their query-URL vector similarity, our approach adaptively controls the output of different lists of related queries in terms of the level of users' satisfaction.

In Chapter 7, we study related keyword suggestion in a scenario where an input query is new to Web logs and the cost of pseudo relevance, e.g., utilizing retrieved Web pages, is prohibitively high for large volumes of queries. We propose a novel two-step approach in this chapter. In the offline model-learning step, we collect an appropriate training data by gaining external knowledge from Web to build a generative probabilistic topic model for Web queries. The training set is carefully selected by considering the topic consistency with queries. Then, for each candidate query, its posterior distribution of hidden topics is inferred from the learned model. In the online query suggestion step, the same inference is done for an input query. We estimate the similarity between the input query and candidate queries based on their corresponding topic distributions. Finally, a suggestion list of candidate queries is ordered according to the similarity scores, which gives the hint to search engine users in rephrasing their query formulation.

Chapter 8 gives a summary of main results on personalized re-ranking and related keyword suggestion. We also have a discussion of the future work and remaining open problems.

1.3 Main Contributions

The main contributions of this thesis can be summarized as follows:

- We utilize the metadata expressing topical categorizations of web pages in Open Directory Project as useful means for automated learning of user interests and user search behaviors. We design various re-rank strategies to re-order search results, which effectively learn user-specific query-dependent personalization preference and significantly improve the accuracy of personalized search over the most existing personalized re-ranking schemes.

- To find related keywords, using terms in a query as a feature space cannot accurately estimate relatedness between Web queries. We study alternative feature spaces to enrich the query, thus better representing its meaning. We focus on improving the performance of the content-ignorant feature space by two methods. One is finding a novel feature space by creating Web communities based on link analysis of Web pages. The other is devising a novel rank mechanism for ordering the related queries based on the merging distances of a hierarchical agglomerative clustering (HAC). We conduct comprehensive experiments on different query suggestion strategies and different feature spaces. The proposed methods can find more semantic related Web queries than the conventional approaches.

- Different from the popular implicit and pseudo relevance based query suggestion, we train a probabilistic topic model to represent queries on a semantic level rather than by lexical occurrence. Our hidden topic analysis based approach can apply the trained model to queries no matter they are in the set of known past queries nor not. In addition, it is not necessary to extract external sources from Web in the online suggestion step, which makes our approach adaptive to use in demanding operational environments, such as large-scale Web search services.

Chapter 2

Related Work

This chapter reviews selected publications related to the topics covered in the rest of the thesis. In Section 2.1, we summarize current research works on personalized search. In Section 2.2, we introduce query suggestion strategies such as related query suggestion and query expansion. Finally, Section 2.3 presents rank aggregation methods.

2.1 Personalized Search

2.1.1 Context Search

Kraft et al. [45] state that the context, in its general form, refers to any additional information associated with the query in the web search field, and also present three different algorithms to implement the contextual search instead of modelling user profiles. Generally speaking, if the context information is provided by an individual user in any form, whether automatically or manually, explicitly or implicitly, search engines can use the context to custom-tailor search results. The process is named as a personalized search.

In this way, such a personalized search could be either server-based or client-based. The system in [28] is an available server-based search engine that unifies a hierarchical web-snippet clustering system with a web interface for the personalized search. Google and Yahoo! also supply personalized search services. With the cost of running a large search engine already very high, however, it is likely that the server-based full-scale personalization is too expensive for the major search engines at present.

On a client-based personalized search, studies [25, 83, 94] focus on capturing all the documents edited or viewed by users through computation-consuming procedures. Allowing

for scalability, the client-based personalized search could learn user contexts more accurately than the server-based personalized search, while it is unavoidable that keeping track of user contexts has to be realized by a middleware in the proxy server or the client. Users, however, may feel unsafe to install such a kind of softwares even if they are guaranteed to be non-invasive, and may intend to enjoy the services provided by search engines instead. Moreover, if a user at home uses her private computer which is different from that in her office, keeping her contexts consistent becomes a problem.

2.1.2 Building User Profile

One important component of personalized search is learning users' interests (preferences). There have been many schemes of building user profiles to figure user preferences from text documents. We notice that most of them model user profiles represented by bags of words without considering term correlations [6, 48, 88, 98]. A kind of a simple ontology is a taxonomic hierarchy, particularly constructed as a tree structure, which has been widely accepted to overcome the drawbacks of the bag of words in [16, 43, 66, 74, 81, 89, 93].

The term *Ontology* is borrowed from philosophy, where ontology is a systematic account of existence. In the field of knowledge sharing, Gruber [35] used ontology to mean an explicit specification of a conceptualization. Furthermore, ontology is often equated with taxonomic hierarchies of classes, but not class definitions and the subsumption relation. Labrou et al. [47] used Yahoo! categories as a simple ontology for document classification. The Open Directory Project (ODP) [1] is a large and comprehensive human-edited hierarchical directory of the Web, and is constructed and maintained by volunteer editors.

Persona [93] presented an interactive query scheme utilizing ODP as Web taxonomy and wrapped a personalization module onto search engine. Schickel-ZuberF et al. [81] scored the similarities between user preferences and concepts based on the structure of ontology. However, the two studies need users to express their preferences explicitly. Speretta et al. [89] created user preferences by classifying the information into an ODP concept hierarchy and then re-ranked search results based on conceptual similarity between page and user preferences. They, however, have not taken into consideration the hierarchical structure of ODP when calculating similarity values. Different from the above works, we not only learn the ontology-based user preferences transparently from the click-through data, but also utilize hierarchical similarity measures to evaluate the similarities between users and search results.

[1] http://dmoz.org

In learning user preferences, some topics will become more interesting to the user, while the user will completely or to varying degrees, lose interest in other topics. Studies [6, 48, 98] suggest that relevance feedback and machine learning techniques show promise in adapting to changes of user interests and reducing user involvements, while still overseeing what users dislike and their interest degradation. In [48] a two-level approach is proposed to learn user profiles for information filtering. While the lower level learns the stationary user preferences, the higher level detects changes of user preferences. In [98] a multiple three-descriptor representation is introduced to learn changes in multiple interest categories, and it also needs positive and negative relevance feedback provided by users explicitly. In Chapter 3 we discuss how to learn user preference automatically without any extra efforts from users.

2.1.3 Personalized Re-ranking

More specifically speaking, there are two kinds of context information we can use to model search experience and capture user search histories. One is short-term context, which emphasizes that the most recent search is most directly close to the user's current information need [60, 82, 83]. Successive searches in a session usually have the same information need. Detecting a session boundary, however, is a difficult task. The other is long-term context, which generally assumes that users will hold their interests over a relatively long time. It means that any search in the past may have some effect on the current search [16, 59, 72, 94]. These studies commonly used all available contexts as a whole to improve the search result quality and ranking. Preliminary discussion on this problem in [92] is in the context of only exploiting long-term search history of users. A unique characteristics of our framework proposed in Chapter 4 is the development of a selective use of personalized search history and a combination of long term and short term user search histories in rank optimization of personalized search.

2.2 Keyword-based Query Suggestion

Commercial search engines give suggestions on the queries input by users, thus assisting them in rephrasing their query formulation to improve search quality, such as *Related search terms* in Google and *Search Assist* in Yahoo!. These services supplied by Google and Yahoo! highlight the importance of query suggestion which is considered an effective assistant in enhancing keyword based queries in search engines and Web search software. The main process of query suggestion consists of two steps: (1) finding the terms from queries or documents that are most similar to the current query, and (2) ranking similar terms and utilizing the ranked similar terms to reformulate the current query, such as

appending terms to the existing list of terms in the query, replacing some terms in the query and so on. Existing query suggestion techniques differ from one another in terms of the methods they use to find similar terms and the techniques they use to rank the similar terms.

2.2.1 Finding Related Keywords

The techniques to find semantically related queries is becoming an increasingly important research topic that attracts considerable attention. Existing techniques differ from one another in terms of how to improve the naïve query term based suggestion which simple thinks that two Web queries are related if they share common terms.

Pseudo relevance feedback is widely used [33, 55, 101]. Glance [33] introduced a software agent that collects queries from previous users, and determined the query similarity based on the Web pages returned by queries, and not the actual terms in the queries themselves. Li et al. [55] devised a novel rank mechanism for ordering the related queries based on the merging distances of a hierarchical agglomerative clustering (HAC).

On the Web, recent studies are interested in using Web logs as an additional source to enrich short Web queries [2, 3, 5, 52, 61, 63, 97, 101]. There are two kinds of feature spaces commonly used in the literature, i.e., content-sensitive and content-ignorant features. Beeferman et al. [5] used single-linkage clustering to cluster related queries based on the common clicked URLs two queries share, a content-ignorant feature space. Wen et al. [97] further proposed three kinds of features to compute query to query relatedness: 1) based on terms of the query, 2) based on common clicked URLs, and 3) based on the distance of the clicked Web pages in a predefined hierarchy. The terms in a short Web query would not give reliable information, while the limitation of URL feature space is that two Web pages with different URLs may be semantically related in contents. The third features in [97] needs a concept taxonomy and requires Web pages to be classified into the taxonomy as well. Such taxonomy is not generally available. Baeza-Yates et al. [2] find related queries based on the content of clicked Web pages using click frequency as a weighting scheme. Their experiments show that using the content information of a Web page (e.g., nouns) is a more accurate query enrichment way to measure query similarity than using the URL of a Web page.

There are some recently proposed methods which went other directions. Some studies viewed Web logs as a set of transactions where a single submits a sequence of related queries in a time interval [12, 31, 84]. They can make meaningful query suggestions by applying traditional data mining techniques, such as association rule mining. These techniques succeed because they mine "wisdom of the crowds" for query understanding.

Chien et al. [14] defined a new measure of the temporal correlation of two queries based on the correlation coefficient of their frequency functions, which requires a time-stamped query stream as input and can be used as a complementary approach to the above discussed methods.

2.2.2 Query-URL Context

The content-based feature space, e.g., terms of a Web page, however, is not applicable, at least in principle, in settings including: non-text pages like multimedia (image) files, Usenet archives, sites with registration requirement, and dynamic pages returned in response to a submitted query and so forth. It is crucial to improve the quality of the URL (content-ignorant) feature space since it is generally available in all types of Web pages.

The query-URL relationship can be represented by a bipartite graph. Finding biclique is a natural way of collecting the most related queries and URLs. A well known problem related to biclique is the maximum clique, which is one of the most widely studied NP-complete problems in the literature [20]. Graph partitioning is an alternative for grouping [76, 104] which is done by cutting the set of vertices into disjoint sets. Beeferman et al. [5] viewed the click-through data as a bipartite graph, and utilized an iterative, agglomerative clustering algorithm to the vertices of the graph for clustering queries and URLs, respectively. Their method just extracted connected components from the entire graph and used $frequency$ to measure the similarity between queries. The limitation of this method is the weakness of selecting queries from the most frequently occurred connected component that contains the input query (keyword list). However, the selected queries may not be the best query suggestion, as the frequency is not always the best descriptor of relatedness because it does not discern the individual targeted queries.

Our QUBiC approach proposed in Chapter 5 can be regarded as a combination of traditional clustering strategy and graph analysis, which makes use of the dendrogram of clustering to reflect the global similarity through a similarity graph and propose a novel tree distance based ranking for query recommendation. SimRank [41] measured the object-to-object relationship by recursively scoring the similarity of their related objects. Sun et al. [90] employed random walk with restarts and graph partitioning to solve two problems, neighborhood formation and anomaly detection. The two studies reflect the global similarity since they work by iteratively transmitting weights through the whole graph. However, they supplied all the users with the same list of related queries in response to an input query. Namely, they ignored the subjectivity of similarity, especially

the users' diverse needs. Our approach can adaptively output the recommendation list of related queries according to users' needs.

In addition, an alternative representation for query-URL data can be given by a contingency matrix whose rows correspond to queries and columns to URLs. This matrix is sparse, since the majority of queries retrieve only a small number of URLs. The elements of the matrix can be set as binary or weighted according to a measure (e.g., each entry is the probability of choosing same query and same URL). In this context, studies [22, 87] cluster documents based on the joint empirical distribution of words and documents.

2.2.3 Query Clustering

Query clustering also helps find related Web queries, which appears to be less explored than clustering Web pages or documents [2, 5, 36, 97]. Wen et al. [97] proposed to cluster similar queries to recommend URLs to frequently asked queries of a search engine. They combined similarities based on query contents and user clicks, and regarded user clicks as an implicit relevance feedback but not the top ranked Web pages. Hansen et al. [36] distilled search-related navigation information from proxy logs to cluster queries. The data they relied on differed from those used in the above other studies. Different from the utilization of clustering in these studies, Chapter 5 makes use of the monotonicity of the HAC strategies to adaptively output the ordered list of related queries. In addition, we examine three URL-based similarity measures analytically and empirically to provide better understanding of the propagation of similarity from query to query by inducing an implicit topical relatedness between queries.

2.2.4 Web Query Expansion

We have introduced suggesting related queries to help users refine their original queries, while finding candidate terms from documents is widely used in query expansion which is also an alternative to revise users' queries. The conventional research efforts on query expansion are classified into three categories according to the information sources they use to find relevant terms for query expansion: 1) corpus-based statistics analysis [96], 2) relevance feedback based recommendation [80], and 3) local context based recommendation [10, 100]. In addition, some researches combined multiple sources of knowledge on term associations [15, 17, 91]. Others used Web logs to bridge the term gap between user-centric query space and author-centric Web page space [19, 107]. Most query expansion techniques suggest terms used extracted from Web pages. However, some terms are difficult to be suggested because of their high document frequencies. We think that if these terms appear in some past queries, we recommend them to users by using the full

term list of a query. Therefore, terms in related queries can also be an effective source of expansion terms. Further experiments on query expansion using related queries would be an interesting topic in our future work.

2.3 Rank Aggregation

Given a number of ranked lists, the task of rank aggregation is to combine them to a broadly acceptable list that minimizes the number of disagreements with the respective lists. This problem has arisen in many fields such as metasearch, combating spams, ensemble clustering, and so on. For example, metasearch deals with the problem of merging the result lists returned by multiple search engines in response to a same query.

Many rank aggregation methods have been proposed in the literature. They can be classified based on whether: 1) they rely on score (value) [64, 77, 85]; 2) they rely on rank (order) [1, 26, 27, 51, 65, 102, 106]; 3) they require training data or not [77]. Renda et al. [77] showed that the performance of some rank-based methods are comparable to score-based methods. Their conclusion is useful in the context of Web search since scores are usually unavailable from search engines. Rank aggregation has proved to a useful and powerful paradigm in several applications including metasearch [1, 26, 65, 77, 106], Web page spam reduction [26], classification [27, 51], database-centric applications [102] and so on. Metasearch is one of the successful applications of order-based aggregation, which submits a query to several search engines assuming that the query is well phrased. Metasearch is orthogonal to our research because we do parallel searches of several queries on a search engine. Iterative filtering meta-search (IFM) [45] generates multiple subqueries by augmenting a user's query with appropriate terms from a set of 200 contexts, and re-ranks the results of subqueries using traditional rank aggregation methods. In our personalized re-ranking, we store the attributes of user's interests and form a broadly acceptable rank list among these attributes by making use of rank aggregation. In addition, rank aggregation is also a popular and effective way to combine the results of different approaches.

Chapter 3

Personalized Re-ranking Using Query and Time Independent Strategy

Augmenting the global ranking based on the linkage structure of the Web by creating personalized view of importance is one of the popular approaches in online information system today for enhancing the search and ranking quality of Web-based services. This chapter introduces an adaptive scheme to learn the changes of user preferences from click-through data, and re-ranking methods to bias the search results of each user. We propose independent models for long-term and short-term user preferences to compose our user profile. Dynamic adaptation strategies are devised to capture the accumulation and degradation changes of user preferences, and adjust the content and the structure of the user profile to these changes. Our score based personalized re-ranking uses click frequency as weight and compute the similarity between a search result and a user profile. We also study rank aggregation methods for personalized re-rank optimization. User profiles store the attributes of user preferences, such as the topics that a user is interested in, the degrees of user interests in these topics, and so forth. We introduce a set of techniques to combine the respective rank lists produced by various attributes of user preferences.

The rest of this chapter is organized as follows. In Section 3.1 we introduce the background of personalized search. In Section 3.2 we describe two independent models, dynamic adaptation strategies for user profiles, score combination as re-ranking, and experimental results. In Section 3.3 describe rank-based rank aggregation, including how to produce user-centered rank lists and fuse them. Finally, we conclude this chapter in Section 3.4.

3.1 Introduction

With the advent of the era of the information explosion, never before have there been so many information sources availably indexed by search engines on the Internet. Ideally, users should be able to take advantage of the wide range of the valuable information while being able to find only those which are appealing to them. On the contrary, it becomes more difficult than ever to obtain desired results due to the ambiguity of users' information needs. Moreover, present search engines generally handle search queries without considering user preferences or contexts in which users submit their search queries. For example, suppose that a database researcher who wants to search for information about a conference on Mobile Data Management and a banker who is interested in searching for the MDM bank, both input "MDM" on Google search engine. Regardless of the different intentions of the two users on the same query, the results turn out to be a multimedia software company, a broadband services company, a national observatory, a conference on mobile data management, and so on. Current search engines prove unfortunately inadequate for this situation. To address this problem, personalized search has recently become an active on-going research field. In this chapter, we focus on studying learning user profiles and utilizing the learned user profiles to re-rank search results.

Studies [16, 74] requires search engine users to explicitly enter their contextual preferences including interest topics, bookmarks, and so forth. These contextual preferences are used to expand user queries or re-rank search results. Forcing users to submit their contextual preferences would be a task that few users would be willing to do. Furthermore, it is very difficult for users to define their own contextual preferences accurately. Much attention has been paid in [73, 83, 89, 94] to learn user preferences transparently without any extra effort from users. These studies place emphasis on modelling user profiles or user representations to indicate user preferences automatically. Moreover, user preferences can be represented by a bag of words or a taxonomic hierarchy. The bag of word representation does not consider term correlations because terms in user preferences are considered in isolation from one another. The taxonomic hierarchy can overcome this drawback and has been widely accepted [16, 81, 89].

Most studies on learning user profiles have deemed user profiles to be static. A related problem occurs when user preferences change over time. For instance, if a user changes her vocation from being an IT specialist to a lawyer, her interests will naturally shift with this change. It becomes important to keep the user profile up-to-date, and for a search engine to adapt accordingly. In addition, a user profile covers both short-term and long-term user preferences, which may increase or reduce respectively and co-relatedly with time. Using one model to represent two differently featured parts of the user profile will be far from perfect. Accordingly, suitable strategies are needed to capture

the accumulation and degradation of changes of user preferences, and then adapt the content and the structure of a user profile to these changes.

After a user profile is learned, we should compute its similarity scores with search results. This similarity computation is called personalized re-ranking. Our rank mechanism is similar to that proposed by [16] in which a semantic similarity measure is introduced with consideration to the hierarchy of the ODP structure. However, the technique proposed in [16] suffers from the problem of requiring users to select topics which best fit their interests from the ODP, and other shortcomings we will address in Section 3.2. This personalized re-ranking strategy is called score combination.

On the other hand, a user profile may contain a number of attributes which describe user interests from their respective viewpoints. In most cases, any individual attribute is deficient in representing user interests accurately. In order to leverage the different ranking lists produced by the different attributes, the rank aggregation should intend to form a single ranking list supported by a broad consensus among these attributes. Merging the values of the attributes in a simply linear combination [16, 89] may result in neglecting their respective characteristics. Moreover, it is important to observe that if the ranking measure is value-based, the ordering implied by the values makes more sense than the actual values themselves [26]. Dwork et al. [26] also developed the theoretical ground work for describing and evaluating rank aggregation methods. Their main work is to effectively combat "spam". We also study the rank aggregation of the attributes of user interests learned from the click-through data to improve the web search. This personalized re-ranking strategy is called rank aggregation.

3.2 Score Combination for Re-ranking

In this section, our work is summarized as follows.

(1) We devise independent models for long-term and short-term user preferences.

(2) Dynamic adaptation strategies for modelling user profiles automatically are proposed. These strategies are based on click-through data while considering the accumulation and degradation changes of user preferences.

(3) When user preferences change, our user profiles, not only in contents, but also in structures, are modified to adapt to the changes.

(4) Finally, we propose a novel rank mechanism by measuring hierarchy semantic similarities between up-to-date user profiles and web pages. About 29.14% average improvement is gained over baselines.

3.2.1 User Profile and Dynamic Adaptation Strategies

As indicated in [98], for user profiles, long-term user preferences generally hold user preferences and the degree of preferences accumulated by experiences over a long time period. Hence it is fairly stable. On the other hand, short-term user preferences are unstable by nature. For instance, interests in current hot topics could change on a day-to-day basis. It is crucial to design a temporal structure for shot-term user preferences. Based on these features, we propose two novel models for long-term and short-term user preferences respectively and discuss them together with the adaptation strategies for their close correlations. Our strategies are in accord with the changes of user preferences in nature.

3.2.1.1 Long-term Model of User Profile

The taxonomic hierarchy for our long-term model is a part of the Google Directory [34]. This part is composed of topics that have only been associated with the clicked search results, instead of the whole Google Directory. And these topics are linked as a tree structure to form our long-term model that is also called the user topic tree (UTT) from now on. In other words, each node in the user topic tree means a topic in the Google Directory. We use search results and web pages interchangeably when referring to the URLs returned from the web search engine on a specific query.

In the Google Directory, each web page is classified into a topic[1]. In the "adding" operation, topics associated with the clicked pages, and not all the search results, are added into the UTT click by click. Moreover, each node in the UTT has a value of the number of times the node has been visited. This value is called the "$TopicCount$", and represents the degree of preferences. The "deleting" operation is effected by the changes of the short-term model. It will be addressed in Section 3.2. Figure 3.1 illustrates the schema of the user topic tree. For example, node C is represented by the $[Internet, 18]$ which means one user has clicked a page associated with the topic "$Internet$" and the user has visited the "$Internet$" 18 times before this search. In our experiments node C is actually stored as the $[\backslash Root \backslash Compuetr \setminus Internet, 18]$ with a full path in the Google Directory.

[1]If necessary, all the symbolic links may be loaded into memory or the shortest distance on the graph is computed.

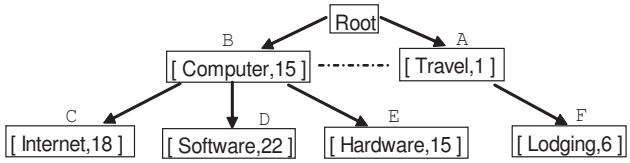

FIGURE 3.1: Schema of long-term user profile

3.2.1.2 Short-term Model of User Profile

Here we frame the Topic-History Buffer (THB) for the short-term model. The THB caches the most recently clicked topics with a fixed size that is determined by the ability of the search engine. We now meet the same problem as the cache in the processor, and that is how to kick off the "old" topics in time to keep up with the changes of short-term user interests. As it is known, in the cache management, there are popular cache replacement algorithms that are all designed for the processor, the web cache and the database disk buffering. No such research could be available in the personalized search, especially in the short-term model of the user profile. Our goal, keeping track of the most recent accesses of search results in the THB, is basically similar to that in the cache management. As a result, the LFU (Least Frequently Used), one of the cache replacement algorithms, is applied to our scheme, which is named the Least Frequent Used Topic Replacement (LFUTR). The details are shown in Table 3.1. The LFUTR reflects the changes of the short-term model, including how to add (line 3 $\sim$ line 6) and replace (line 10 $\sim$ line 11) topics in the THB.

From Figure 3.1 and the LFUTR algorithm in Table 3.1, our dynamic adaptation strategies maintain user profiles such that the short-term model is updated by the LRCTR (line 1 $\sim$ line 14, while the degree of interests in the long-term model could be degraded (line 12) when the topic in the THB is replaced, and may be accumulated when the user clicks the web page ("adding" operation). On the other hand, if the user accesses the web page whose associated topic is not in the current UTT, the new node could be added into the tree ("adding" operation). From line 15 to line 17, if the "*TopicCount*" of one node becomes zero, the node would be deleted from the tree. This procedure is called the "deleting" operation for the long-term model. The "adding" and "deleting" operations dynamically adapt the structure of the long-term model to the user click behaviors. Although we design independent models for short-term and long-term user interests, our strategies ensure that the inherent correlations between them are not ignored, and that the changes of the short-term model have an even influence on the long-term model. Here, the meaning of "even" is that we degrade the "*TopicCount*" not on an hour-to-hour or a day-to-day basis, only after a period of time during which the user has not accessed the *topic* in the whole search process.

Table 3.1: LFUTR algorithm

```
INPUT:  Current short-term model, Current long-term model, Search results
OUTPUT: Updated user profile
Parameters:
BTopicCount=Vector of the number of clicked times for topics in the THB
TopicCount=Vector of the number of clicked times for topics in the UTT
BufferTopics=Vector of topics in the THB
Results=Vector of web pages returned by a search engine
UserTopics=Vector of topics(nodes) in the UTT
For i=1 to Size(Results)
Begin Loop
If the topic of Result[i] is IN the THB
   BTopicCount[Results[i]]++;
Else If THB is NOT FULL
        Add the clicked topic into the end of the THB;
     Else
        Begin
        For j=1 to Size (THB)
          BufferTopics[k] ← Find one topic in the BufferTopics
          with the Minimum BTopicCount[j];
          Replace the BufferTopics[k] with the topic of Results[i];
          TopicCount[BufferTopics[k]]- -;
        End
End loop
For t=1 to Size (UserTopics)
If TopicCount[t] ==0
   Clear the UserTopics[t] out from the UTT
```

3.2.2 Rank Mechanisms and Evaluation for Personalized Search

3.2.2.1 Distance metrics

The tree distance which we deal with, is the distance between each search result and the user topic tree, as described in [16]. The search result with the shorter distance, meaning the higher similarity to user preferences, should be put in the topmost position of the ranking list. For each search result, there is an associated node in the Google Directory. The user topic tree is also composed of nodes. The distance computation is actually how the distance between two nodes in the tree structure is measured.

Chirita et al. [16] point out that the main drawback of the naïve tree distance is that it overlooks the depth of the subsumer (the deepest node common to two nodes). With the help of Figure 3.1, let us explain the problem clearly. $sub_{i,j}$ represents the subsumer of the node i and the node j. $Edges(i, sub_{i,j})$ represents the number of edges between

the node i and the node $sub_{i,j}$. The naïve distance is defined as

$$Distance(i,j) = Edges(i, sub_{i,j}) + Edges(j, sub_{i,j})\,. \tag{3.1}$$

$Distance(A,B)$ is 2, which is the same as $Distance(C,D)$, making it difficult to re-order search results by Equation (3.1).

3.2.2.2 Hierarchy Semantic Similarity

Li et al. [57] takes the depth of the subsumer h and the naïve distance between two nodes l into the calculation. α and β are the parameters scaling the contribution of the naïve distance and the depth respectively. The semantic similarity is defined as

$$Sim(i,j) = e^{-\alpha \cdot l} \cdot \frac{e^{\beta \cdot h} - e^{-\beta \cdot h}}{e^{\beta \cdot h} + e^{-\beta \cdot h}}\,, \quad \alpha \geq 0\,, \beta > 0\,. \tag{3.2}$$

Their experiment results show that the optimized values of the two parameters are, α=0.2 and β=0.6. For example, $Sim(A,B)$ is unequal to $Sim(C,D)$ based on Equation (3.2). Because the subsumer of A and B, i.e., "*Root*", is in the different level from the subsumer of C and D, i.e., "*Computer*". However, Equation (3.2) only solves problem partially. Let us see another example. Due to the same value (i.e., 3) between $Distance(A,C)$ and $Distance(B,F)$, and the same subsumer (i.e., "Root") between the pairs (A, C) and (B,F), $Sim(A,C)$ is equal to $Sim(B,F)$.

Under this situation, Chirita et al. [16] separate l into l_1 and l_2, and then gives different weights to the two variables through the parameter δ defined as

$$Sim^*(i,j) = ((1-\delta) \cdot e^{-\alpha \cdot l_1} + \delta \cdot e^{-\alpha \cdot l_2}) \cdot \frac{e^{\beta \cdot h} - e^{-\beta \cdot h}}{e^{\beta \cdot h} + e^{-\beta \cdot h}}\,. \tag{3.3}$$

Equation (3.3) can work well for common cases. However, we find that the parameter δ is sensitive to the semantic meanings between the two topics, as illustrated in [16]. Furthermore, even if we compute the similarity by Equation (3.3), $Sim(C,D)$ is still equal to $Sim(E,D)$ because of the same value between l_1 and l_2. In our system, we extend Equation (3.2) in another way, as the "*TopicCount*" has much better effect on the overall performance than the weak parameter δ. Comparative experiments are in Section 5.

3.2.2.3 Our Rank Mechanism

When a user submits a query to the search engine, the search results are re-ranked by our semantic similarity defined as

$$CSim(i,j) = WT(i) * Sim(i,j)\,, \tag{3.4}$$

the degree by which the search result is similar to the user profile. i is a node in the user topic tree ($i = 1, 2, \cdots, size(UserTopics)$). j is the associated node with a search result in the Google Directory ($j = 1, 2, \cdots, size(Results)$). $WT(i)$ is defined as $TopicCount(i)/\sum_{i=1}^{size(UserTopics)} TopicCount(i)$, weighing the degree of preferences of a node in the user topic tree.

The larger the WT is, the more interested the user is in one topic. For one search result, the number of the $CSim$ values is $size(UserTopics)$ in Equation (3.4). One user topic tree represents one user. We define the semantic similarity between one search result and the user topic tree as the maximum value among all the values ($i = 1, 2, \cdots, size(UserTopics)$) expressed as

$$CSim^*(User, j) = Max(WT(i) * Sim(i,j))\,. \tag{3.5}$$

To keep our rank mechanism from missing the high quality pages in Google, Equation (3.5) is integrated with PageRank [70] as

$$FinalRank(User, j) = (1-\gamma)CSim^*(User, j) + \gamma * PageRank(j)\,. \tag{3.6}$$

Here γ is a parameter in [0,1] which blends the two ranking measures. The user could vary the value of γ to merge our rank mechanism and PageRank in different weights. In our experiments, γ is set to 0.5, which gives equal weight to the two measures.

3.2.3 Experiments

3.2.3.1 Experimental Setup

Our rank mechanism could be combined with any search engine. In this study we choose the Google Directory Search [2] as our baseline in that Google applies its patented PageRank technology on the Google Directory to rank the sites based on their importance. It is convenient for us to combine and evaluate our rank mechanism with Google. The necessary steps are depicted in Table 3.2. Main modules in the experiments are listed

[2] http://directory.google.com/

Table 3.2: Procedure of evaluation experiments

1.	Issuing the query submitted by an online user through the Google API module ;
2.	Re-ranking search results by our rank mechanism based on the current user profile and then going into the Log module;
3.	Adapting the user profile to click-history data provided by the Log module through our strategies:
4.	For the long-term model updating the structure and the degree of preferences by the "adding" operation;
5.	For the short-term model, updating web pages in the PHB by the LFUPR algorithm;
6.	If needed, degrading the long-term model according to the changes of the short-term model by the "deleting" operation.
7.	Waiting until the online user submits a new query, and then going to 1.

as follows.

(1) Google API module: Given a query, we are offered titles, snippets, and page-associated Google directories beside the URLs of web pages by the Google API [3]. Here a Google directory is regarded as a topic in the user topic tree.

(2) Log module: We monitor user click behaviors, recording the query time, clicked search results, associated topics.

(3) User profile: It has been described in Section 3.2.1.

In our experiments, due to the large size of the whole Google Directory, only the top 4 levels are encoded into the user topic tree. The size of the PHB is 20 pages. Ideally if we could cache all the clicked web pages in the PHB and utilize the whole levels of the Googe Directory, it would be much easier to personalize a search.

3.2.3.2 Dataset

For each search, the Google API module got the order of the top 20 Google results due to the limited number of the Google API licenses we have. We randomized the order of the results before returning the 20 results to the user at run-time. For evaluation, 12 subjects are invited to search through our system. The 12 subjects are graduate students (5 females and 7 males) researching in several fields, i.e., computer, chemistry, food engineering, electrical engineering, art design, medical, math, architecture, and law. These subjects are divided into three types:

- Clear User, searching on queries that usually have one meaning,

[3] http://code.google.com/apis/soapsearch/

- Semi-ambiguous User, searching on queries that have two or three meanings,
- Ambiguous User, searching on queries that have more than three meanings.

Our search interface was available on the Internet, and convenient for the subjects to access it at any time. They were asked to query topics closely related to their interests and majors. In the first four days, subjects input the queries on their majors, and then in the next three days the queries on their hobbies were searched. Finally, in the last three days, the subjects were required to repeat some queries done before. This repeated procedure gave a clear performance comparison between the current and earlier systems, as user profiles were updated search by search. After the data were collected over a ten-day period (From October 23nd, 2006, to November 1st, 2006), we got a log of about 300 queries averaging 25 queries per subject and about 1200 records of the pages the subjects clicked in total.

3.2.3.3 Evaluation Metrics

As our goal is to model user profile from click-history data and use it to realize personalized search, we plan to consider the following evaluation metrics:

(1) **Error** evaluates the accuracy of learned user profiles. It is natural to evaluate our user profiles by computing the difference between the real user topic tree and the modelled user topic tree. Equation (3.2) suitable for this task and the relative error between the two user profiles is defined as:

$$Error(M) = \frac{|Sim(M,R) - Sim(R,R)|}{Sim(R,R)}\,, \tag{3.7}$$

where $Sim(M,R)$ is denoted by $\sum_{j=1}^{K} Max(Sim(j,i))$. R means the vector of topics in the real user topic tree. M means the vector of topics in the modelled user topic tree. i is a node in R ($i = 1, 2, \cdots, N \rightarrow size(R)$). j is a node in M ($j = 1, 2, \cdots, K \rightarrow size(M)$). A smaller value of $Error(M)$ means a higher accuracy of our modelled user profile.

(2) **AvgRank** indicates the average rank of search results. Whether a personalized system is successful or not is determined by the user satisfaction. An effective rank mechanism should place relevant search results close to the top of the rank list. We ask the subjects to select the search results they consider relevant to their preferences. The measure is defined as follows:

$$AvgRank(q) = \sum_{p \in S} R(p)/|S|\,. \tag{3.8}$$

FIGURE 3.2: Structure of user profile after 10 days

Here S denotes the set of search results selected by a subject for query q, $R(p)$ is the position of p in the result list, and $|S|$ is the cardinality of the set S. A smaller $AvgRank$ represents a better quality.

3.2.3.4 Experimental Results

An Example of User Profile. The right frame of Fig. 3.2 illustrates an example of one user profile structure in the last day with the user showing interests in architecture, design, photography and so on. The left frame is our search interface.

Results of Accuracy of User Profile. From Figure 3.3, we see that as the days went on, the relative errors of our user profiles generally kept decreasing. In the last three days they even apparently stopped decreasing. The trend was expected because the subjects were asked to repeat some queries done earlier for comparison. Without a new query for a search, we are not able to learn more about the user preferences. Moreover, relative errors got even slightly larger on these days. Because the subjects might click pages different from those of the early search on the same query. This further indicates the importance of adaptation strategies to learn the changes in user preferences. Figure 3.3 also shows that it is easier and quicker to learn the user profile of a "Clear User" than that of a "Semi-ambiguous User" and slowest to learn the user profile of an "Ambiguous User". For example, when day=4, Error(Clear User)=0.3, Error(Semi-ambiguous User)=0.6, and Error(Ambiguous User)=0.8. Although the learning procedure of the "Ambiguous User" is slower than the other two kinds of users, as long as its user profile is converged

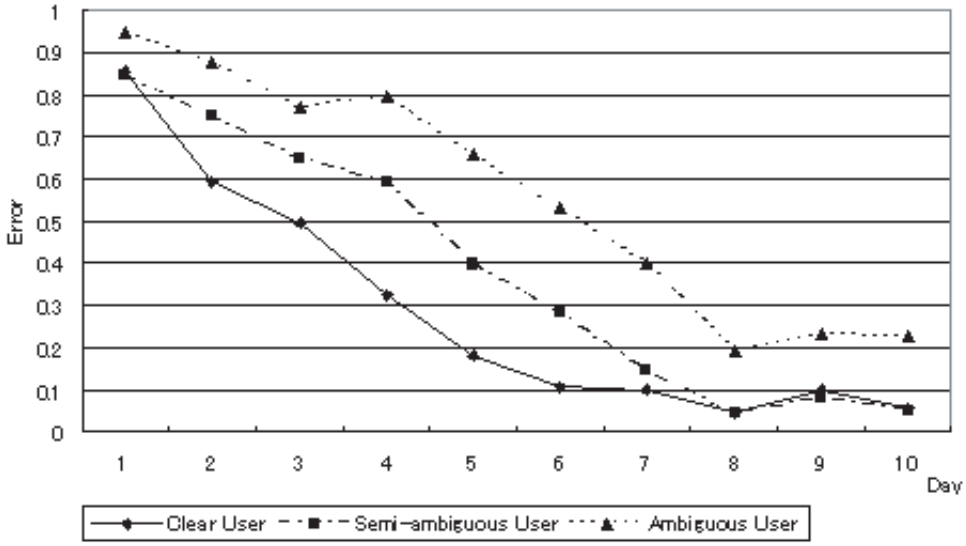

FIGURE 3.3: Accuracy of user profile

relatively, it yields the best improvement in terms of quality among all the three kinds of users.

Results of Quality of Personalized Search System. Now, we compare the performance improvements of the following three ranking mechanisms:

- Google Directory Search (GDS), using the Google API,
- Personalized Google Directory Search (PGDS3), combing Equation (3.3) PageRank,
- Personalized Google Directory Search (PGDS6), using Equation (3.6).

Evaluated by Equation (3.8), how they performed day by day is shown in Figure 3.4. By using the GDS as a baseline, the performance improvement of our PGDS6 in Figure 3.4(b) is 42.37 %, which outperforms those in both Figure 3.4(a) (i.e., 28.86%) and Figure 3.4(c)(i.e., 16.27%). The little improvement in Figure 3.4(c) indicates that GDS has done well with the "Clear User". However, for the "Semi-ambiguous User" and the "Ambiguous User", the significant improvements in Figure 3.4(a) and Figure 3.4(b)illustrates that GDS works worse than our strategies.

Figure 3.4(d) illustrates the average improvement over all users. As a result of requiring the subjects to change queries from their majors to hobbies, we see that from the fourth day to the fifth day, the values of AveRank experience a sudden increase. But after three days on learning the changes, our PGDS6 shows better results than the GDS and the PGDS3. More accurately, compared with the GDS, our PGDS6 outperforms the PGDS3

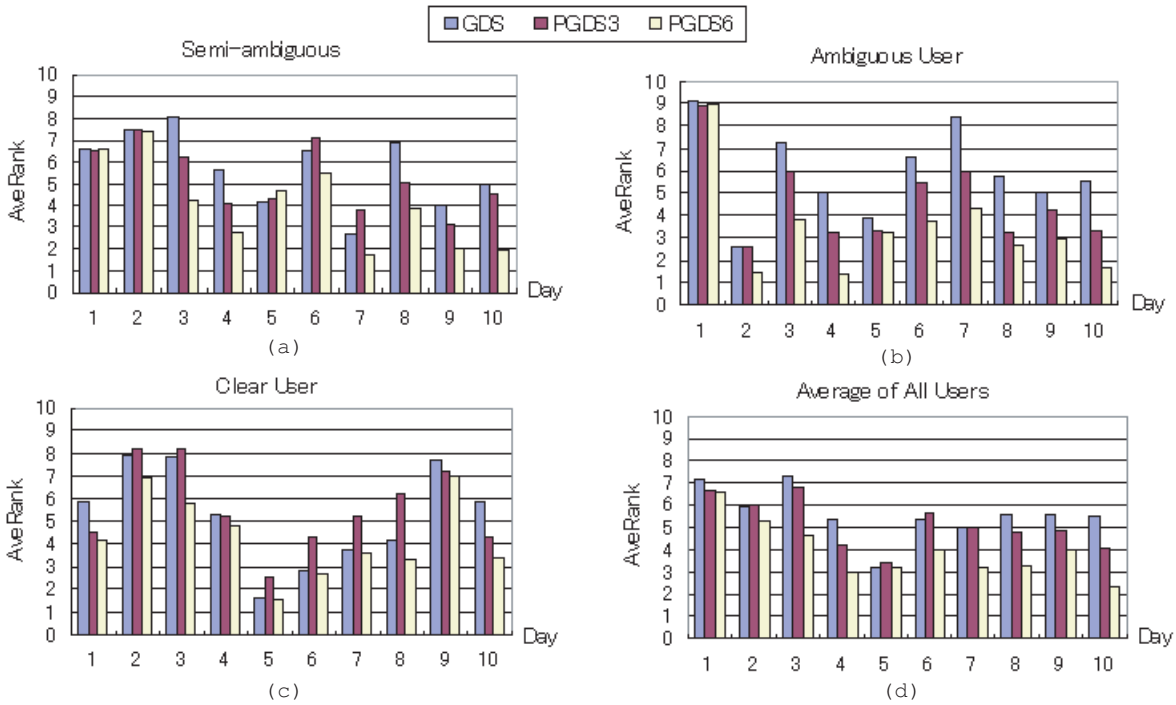

FIGURE 3.4: Quality of personalized search system (Lower is better)

with a 60% improvement for the tenth day, while for the fifth day the improvement is only around 2%. This difference demonstrates that the changes of user preferences will lower the improvement that our strategy could achieve. Nevertheless, our rank mechanism still greatly improves over the GDS and the PGDS3 overall. The average improvements of our PGDS6 and the PGDS3 over the GDS, are 29.14% and 7.36% respectively.

3.3 Rank Aggregation for Re-ranking

User preferences consist of a number of attributes. Each attribute describes a user's favorite in different aspects. In most cases, any individual attribute is deficient in accurately representing user preferences. Combining user knowledge depicted by each attribute can help us understand user preferences well, which finally results in an effective rank mechanism in the Web search. To leverage the rankings produced by the different attributes, rank aggregation intends to form a single rank list supported by a broad consensus among these attributes. There are two approaches: score-based and rank-based. The score-based rank aggregation merges the values of the attributes [16, 89]. In Section 3.2, our personalized re-ranking is score-based aggregation. However, it is important to observe that if the rank mechanism is score-based, the sequence implied by the scores makes it more meaningful than the actual scores themselves. On the other hand, the rank-based rank aggregation fuses the rank lists produced by the values of

the attributes and has been studied and employed in many applications in the last half century [26, 77, 106]. Renda et al. [77] compared rank and score based methods without training data in the context of metasearch, and showed that Markov chain rank-based methods compete with score-based methods. Dwork et al. [26] developed the theoretical groundwork for describing and evaluating rank aggregation methods. Their main point is to effectively combat *spam*. In this chapter we propose methods to effectively improve the Web search in a context-aware manner. From this point of view, our work is quiet different from traditional applications like metasearch, ensemble clustering, and database. Especially, we take into account the respective attributes of user preferences learned from click-through data.

In this section, our work is summarized as follows:

1) We study rank-based rank aggregation in terms of individual user preferences, so it is not a one-size-fits-all method for all the users. A set of techniques originating from social choice theory and graph theory are presented to aggregate the user-centered rank lists plus the original list of a search engine (i.e., Google).

2) Experimental results on a real click-through data show that some proposed methods greatly improved the quality of the Web search. One of Footrule distance based methods showed the best results.

3) We did an overall experimental analysis on a number of hierarchical semantic similarity measures. We found that the rank-based methods are much more robust to the selection of measures than the score-based one. Moreover, Equation 3.3 in Section 3.2, recommended by [16, 57], is not always the best choice in our rank-based methods.

3.3.1 Rank Aggregation

Given a query, a user will click some search results returned by the Google Directory search. Firstly, we record the topics associated with these clicked results and the number of times clicked by the user. Then, when the user submits a query again, we make use of the user information to produce several new rank lists. Lastly, the following proposed methods are applied to form a new rank list instead of the original provided by Google. For an effective rank mechanism, the more similar a search result is to user preferences, the higher position it will be put in the final rank list. In the following part, we will discuss how to get the respective rank lists according to the learned use preferences and introduce the proposed rank aggregation methods.

3.3.1.1 Hierarchical Similarity Measures

Our user preferences are structured as a semantic hierarchy shown in Figure 3.1, so that hierarchical similarity measures are needed to assess the relatedness between user preferences and search results. Li et al. [57] have done comprehensive experiments on a series of hierarchical similarity measures and reported the optimal values of the parameters in these measure definitions. They measure the semantic similarity between words while our work aims at improving the Web search. Moreover, we choose five content-ignorant measures because we want to see how much we can benefit from the hierarchical structure. The five measures are defined as

$$S_1(i,j) = 2 \cdot M - l\,, \tag{3.9}$$

$$S_2(i,j) = \alpha S_1(i,j) + \beta h \qquad (\alpha = 0.05, \beta = 1)\,, \tag{3.10}$$

$$S_3(i,j) = e^{-\alpha \cdot l} \qquad (\alpha = 0.25)\,, \tag{3.11}$$

$$S_4(i,j) = \frac{e^{\beta \cdot h} - e^{-\beta \cdot h}}{e^{\beta \cdot h} + e^{-\beta \cdot h}} \qquad (\beta = 0.15)\,, \tag{3.12}$$

$$S_5(i,j) = e^{-\alpha \cdot l} \cdot \frac{e^{\beta \cdot h} - e^{-\beta \cdot h}}{e^{\beta \cdot h} + e^{-\beta \cdot h}} \quad (\alpha = 0.2, \beta = 0.6)\,, \tag{3.13}$$

where h means the depth of the subsumer (the deepest node common to two nodes), l is the naïve distance (the number of edges or the shortest path length between two nodes), i and j are nodes (topics) in Figure 3.1, and M is the maximum depth of topic directory possessed by user preferences. The values in parentheses are optimal values of parameters. More details can be found in [57].

3.3.1.2 Rank Lists Produced by Ontology-Based User Preferences

The above five content-ignorant measures can evaluate the hierarchical similarity between search results and user preferences. The degree of user preferences has effects on the similarity as well. There are various ways of combining the two kinds of similarity scores dependent on applications. Their product is commonly used in classic IR like our previous work in Section 3.2. However, the ranking implied by the scores has more sense than the actual scores themselves in some cases. In the following discussion we calculate two user-centered rank lists plus the result list returned by Google for rank-based fusion, as distinguished from the traditional score-based combination.

(1) **Hierarchical Semantic Similarity** User preferences include a number of topics (nodes) in Figure 3.1. We further define the semantic similarity between one search result and one user as the maximum value among all the values computed by any one of Equations 3.9-3.13. The search results then are re-ranked and form a rank

list in order of one attribute of user preferences (i.e., the topics a user is interested in). These formulas are popularly used to assess the similarity between words or concepts. The priori work [16], however, just selected one of them without analyzing their differences.

The five measures have their own features from their definitions. For example, compared to Equation 3.9, Equation 3.11 also uses the naïve distance alone, but makes use of a nonlinear function. Equation 3.10 is a linear combination of the naïve distance and the depth. Different from Equation 3.10, Equation 3.13 transfers the naïve distance and the depth by a nonlinear function, respectively, and then combines them by multiplication. Equation 3.12 is the transformation of the depth of the subsumer through a nonlinear function. Based on these differences, we think that it is necessary to experimentally compare their performances when they are applied in the context of the Web search and no priori work has done it. The experimental results are reported later.

(2) **Degree of User Interests** We find that Equations 3.9-3.13 are not round in re-ordering search results. With the help of Figure 3.1, let us explain the problem clearly. The naïve distance between node A and node C (i.e., 3) is the same as that between node B and node F, and the subsumer of A and C (i.e., "root") is the same as that of B and F as well. As a result, computed by any equation from Equations 3.9-3.13, the similarity score between A and C is equal to that between B and F. In this situation, these measures cannot order the two pairs. Our solution for this problem is intuitive that the degree of the user interests in a topic (node) can alleviate this problem. The more times a user clicks one topic, the more interested the user is in it. The user's clicked times can produce a complementary rank list of search results.

(3) **Google List** Google applies its patented PageRank technology on the Google Directory to rank the sites based on their importance. To keep our rank aggregation from missing the high quality Web pages in Google, we also consider the original rank list of Google Directory Search. As we know, there is a PageRank value accompanied with each search result, representing the popularity or authority of results. It certainly could be used to weigh the topics associated with results. Unfortunately, these values are not publicly available for the present, but the ordering of search results can be easily obtained. From this point of view, our rank-based aggregation is suitable in this situation since it is exactly good at processing rank lists. Certainly, it is reasonable for us to guess the approximate values of PageRank if we favor the score-based combination, but this topic is out of the scope of this thesis. In our methods the original rank lists as inputs can intactly and unbiasedly reflect Google's standpoint.

3.3.1.3 Rank Aggregation Methods

We study the problem of combining sets of rank lists from various attributes of user preferences into a single rank list. Voting provides us with a traditional class of algorithms to determine the aggregated rank list. The most common voting theory, named after its creator, is known as Borda's rule [9] as a voting method to elect members to the Academy of Science in Paris. In the context of social choice, we argue that the majority opinion is the truth, or at least the closest that we can come to determining it [103]. Voting can then be described as a method of instilling an objective ranking of the alternatives from a set of subjective preferences. However, the problem with Borda's rule is that it does not optimize any criterion. Dwork et al. [26] recognized that the Kemeny optimal ranking that is an NP-hard problem if the number of rank lists is larger than four. This motivates the problem of finding a ranking that approximately minimizes the number of disagreements with the given inputs. We make use of Footrule distances [23] to weigh edges in a bipartite graph and then find a minimum cost matching. This method was proved in [26] that it can approximate the optimal ranking.

Modified Borda's Rule We simply review Borda's rule. It is a single winner election method in which voters rank candidates in order of preferences. The winner of an election is determined by giving each candidate a certain number of points corresponding to the position in which s/he is ranked by each voter. Once all points have been counted, the candidate with the most points is the winner.

Our idea is that we treat each attribute of user preferences as a voter. It means that each attribute re-orders the search results in the same way as each voter selects a list of candidates. Let $A = a_1, a_2, \cdots, a_m$ be the set of positions in the rank list, and let the attributes of user preferences plus the result list of Google be named by elements of n (i.e., n voters in an election). We shall assume for the present that every element of n can be expressed by a linear order in the position set A. We denote a linear order by a sequence $A_i = a_{i_1}, a_{i_2}, \cdots, a_{i_m}$ where for $j < k$, a_{i_j} is preferred to a_{i_k}.

For each voter, the ranked results should be given some points. The closer a search result is to the top of the list, the more points it will be given. Especially in the context of the Web search, the top search results have much higher possibility to be clicked than others. Most Web search users just browse the top 10 or 20 results. If they do not find the desired information, they will modify their queries to start a new search, instead of continuing checking the results. Therefore, modified Borda's rule is applied here. The voter awards the first-ranked candidate with one point (i.e., 1). The second-ranked candidate receives half of a point (i.e., 1/2), the third-ranked candidate receives

on a third (i.e., 1/3), etc. This kind of point distribution gives more weights to the top results.

When all elements of n have been counted, and each A_i can be thought of as a position vector, we sort the search results by several formulas, defined as

$$L_1(a_k) = \sum_{i=1}^{n} 1/a_{i_k}\,, \quad L_2(a_k) = \sqrt{\sum_{i=1}^{n}(1/a_{i_k})^2}\,, \tag{3.14}$$

$$GM(a_k) = (\prod_{i=1}^{n} 1/a_{i_k})^{1/n}\,. \tag{3.15}$$

Equation 3.14 represents the L_1 norm and the L_2 norm of these position vectors, and the geometric mean of the n points is expressed in Equation 3.15. We take into consideration the median of the n points as well (i.e., the center point if there is an odd number of points, or the value of the sum of the two middle points divided by 2 if there is an even number of points), Borda's rule is commonly classified as a positional voting system because from each voter, candidates receive a certain number of points. Computationally it is very easy, as it can be implemented in linear time.

Bipartite Graph Borda's rule does not assure us that it can find the optimal rank list because it does not optimize any criterion. A graph theory based method is proposed here, to approximate the optimal ranking. We define a weighted balanced bipartite graph $G = (V_1 \cup V_2, W)$. $V_1 = r_1, r_2, \cdots, r_m$ is a set of search results to be ranked. $V_2 = p_1, p_2, \cdots, p_m$ is the m available positions in the rank list. For any two vertices $r \in V_1$ and $p \in V_2$, rp is an edge in G; thus G is also a complete bipartite graph. The weight $W(r, p) =$ is the total distance of a ranking value that places r at position p. The task of rank aggregation is to minimizes the number of disagreements with the respective lists. Therefore, if all the search results are put in proper positions, the total distance (i.e., the number of disagreements) should be the smallest. Now we meet two difficulties in achieving this goal. One is how to compute the distance. The other one is what kind of approaches can minimize the distance.

To weigh the edges in G, according to Diaconis et al. [23], the two distance measures that we consider are:

$$Footrule\_D(\pi, \sigma) = \sum_{i=1}^{n} \mid \pi(i) - \sigma(i) \mid, \; Footrule\_S(\pi, \sigma) = \sum_{i=1}^{n}(\pi(i) - \sigma(i))^2\,, \tag{3.16}$$

where π and σ are regarded as rank lists. Diaconis et al. [23] also suggest two other measures. One roughly seems similar to $Footrule\_D$, and the other is unsuitable for general use, having very small variance about a mean that is very close to its maximum value. Therefore, we choose $Footrule\_D$ and $Footrule\_S$ here. We then adjust the two

measures to compute the total distance that is the weight in an edge, now defined as $\sum_i^n \mid A_i(r) - p \mid$ or $\sum_i^n (A_i(r) - p)^2$.

Minimizing the total distance to n could be solved by the well-known Hungarian algorithm that finds a minimum cost perfect matching in the bipartite graph. A matching in a graph is a set of edges where no two of which share an endpoint. The Hungarian algorithm models the problem as a $m \times m$ cost matrix, where each element represents the cost of assigning the i_{th} search result to the j_{th} position and its time complexity is $O(m^3)$. The most similar work to ours is Dwork et al. [26] who only used $Footrule\_D$ as the distance measure. However, our experiments compared the two measures and observed that $Footrule\_D$ performed the worst among all the methods, even inferior to the score-based method. The largest improvement is reached by $Footrule\_S$. In addition, their main application is to effectively combat *spam* while we study the rank aggregation in terms of user preferences to improve the Web search.

3.3.2 Experiments

3.3.2.1 Evaluation Metrics

As our goal is to model user profile from click-history data and use it to realize personalized search, we plan to consider the evaluation metric $AvgRank$ in Equation 3.8 and DCG.

DCG [40] gives more weight to highly ranked search results, defined as follows:

$$DCG(i) = \begin{cases} G(1) & \text{if } i = 1 \\ DCG(i-1) + G(i)/log(i) & \text{otherwise} \end{cases} \tag{3.17}$$

Note that we must not apply the logarithm-based discount at rank 1 because $log1 = 0$. By averaging over a set of test queries, the average performance of our methods can be analyzed. In the experiments, we used $G(i) = 2$ for highly relevant Web pages, $G(i) = 1$ for relevant Web pages, and $G(i) = 0$ for non-relevant search results. A larger DCG means a better quality.

3.3.2.2 Experimental Results

Depth of Topic Directory for User Preferences The topics in the different depths of the Google Directory compose the user preferences. Consequently, the first step is to determine how deep the depth of topics is when modeling user preferences. To investigate this question, we did a preliminary performance analysis on different depths. The top 20 search results returned by the Google API were manually judged as relevant or not

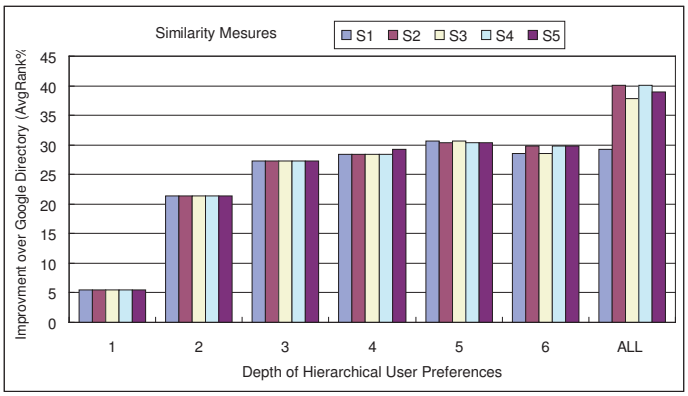

FIGURE 3.5: Depth of Hierarchical User Preferences

by the subjects. The re-ranking mechanism is addressed in [56]. Measured by *AvgRank* in Equation 3.8, Figure 3.5 illustrates the improvement over Google Directory Search per similarity measure versus the depths considered in learning user preferences.

It shows that the deeper the topic directory we process, the bigger improvement is generally reached. If our algorithm stores the whole topic directory, the biggest improvement is over 40%. In addition, we observed that when the depth is set to 1 (2 or 3), the five similarity measures performed almost the same. The reason is that in our dataset, most of the relevant and non-relevant search results share the same subsumer in a very low depth of the hierarchy. We need to store the deeper topic directory to tell the relevant results from the non-relevant ones. Furthermore, from Figure 3.5 when the depth of topic directory increases to 3, the improvement is big, from 5% to above 25%. However, when the depth is increased continually from 3 to 6, the improvement changes slowly. Due to this observation and the large size of the whole Google Directory [4], only the top 4 topic directory is encoded into the user preferences in the following experiments, which is a trade-off between accuracy and storage memory.

When focusing on the performances of the five measures, we observed that S5 (Equation 3.13) is not always the best point. In some deep depths such as 5, 6, etc., S5 is no better than S2 (Equation 3.10) and S4 (Equation 3.12) in quality. Because the two measures, S2 and S4, give much more weight on the depth of the subsumer than the length, the observation also points out that processing our dataset needs deeper topic information from the Google Directory.

Method Selection for Rank Aggregation Measured by DCG in Equation 3.17, we compared the qualities of our techniques and a score-based linear combination (SLC) of measures addressed in [16]. The click-through data produced by repeated queries

[4]Google uses ODP as basis for its Google Directory service, and ODP has more than 590,000 categories

TABLE 3.3: Quality of rank aggregation methods

	Methods	DCG	Improvement
	SLC	1.80557	—
Borda	L_1 norm	1.93948	7.42%
	L_2 norm	1.95184	8.10%
	Median	1.97047	9.13%
	GM	1.98762	10.1%
Footrule	Footrule_D	1.88534	4.42 %
	Footrule_S	2.07534	14.9%

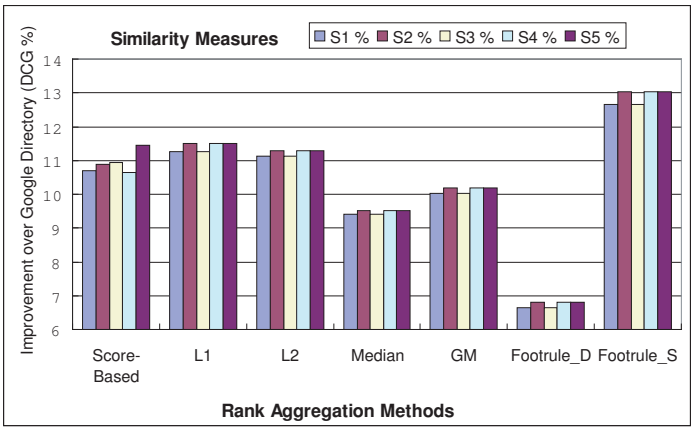

FIGURE 3.6: Similarity measure selection for rank aggregation

are used for this evaluation. This part of data can show us the performances of different rank aggregation methods well without the influences of updating user preferences. The reason is that user preferences are updated query by query, and in the repeating procedure they will be in a relatively stableness. The results of the average improvements over all subjects are illustrated in Table 3.3. Our rank aggregations yield better search results compared with SLC (the largest improvement is 14.9% produced by *Footrule_S*). Although Borda's Rule neither optimizes any criterion nor satisfies the Condorcet property [103], this kind of method outperformed the score-based combination. The Hungarian algorithm based on the Footrule distance that finds a minimum cost perfect matching in the bipartite graph showed the best results obtained by the distance measure *Footrule_S* in Equation 3.16.

Similarity Measure Selection Figure 3.6 illustrates the performances of the five measures defined in Equations 3.9- 3.13. The *Score-Based* method in the figure is the same as that in [56]. The experimental results of this evaluation are different from those in Table 3.3 since the degree of user preferences in one topic is included in the *Score-Based* method and the whole click-through data are used in this evaluation. From this figure, we know that S2, S4, and S5 perform similarly, while S1 and S3 perform similarly as well. The reason is that the former three measures give much more weight on depth than

length, and the latter two measures only consider length. Given the same length and depth, the five measures will compute different values due to different transformation functions. Thus the score-based method is easily influenced by the selected function. On the other hand, the rank-based methods are robust. Even the transformation function is different, as long as the measures take into account the same information, they will produce similar performance. Moreover, in the rank-based methods, S2, S4, and S5 performed slightly better than S1 and S3, which tells us that the depth of subsumbers carry more useful information than the naïve distance in our dataset. Note that S4 uses the depth alone, but competes with S2 and S5. In the score-based method, however, S5 is the winner, much better than the other measures.

For all the measures, the highest improvement is about 13%, and was still produced by $Footrule\_S$. L_1 norm, L_2 norm, and $Footrule\_S$ performed better than *Score-Based*, while the qualities of $Median$ and $Footrule\_D$ are inferior to that of *Score-Based*.

3.4 Summary

In this chapter we introduced how to capture the changes of user preferences from click-through data and how to use the user preferences to re-rank the search results, thus creating personalized views of the web. First, we designed independent models for short-term and long-term user preferences to consist of a user profile. Then, we adapted the user profile, including the content and the structure, to the accumulation and degradation changes of user preferences by our dynamic strategies. Finally, we proposed two kinds of rank mechanism to re-rank search results. One is score based re-ranking and the other is rank based re-ranking. Experimental results on real data demonstrate that our dynamic adaptation strategies are effective and our personalized search system performs better than the selected rank mechanisms, especially for the "Semi-ambiguous User" and the "Ambiguous User". Moreover, we proposed a set of techniques for rank aggregation. We observed that some rank-based aggregation methods performed better than the *Socre-Based* method and the $Footrule\_S$ method performed best in our evaluation. We compared the performances of five similarity measures. If the measures utilize similar information from users, they will perform similarly in rank aggregation regardless of what kind of transformation functions is being used. But the score-based combination is sensitive to the selected function.

The solutions proposed in this chapter [53, 54, 56] utilized the entire recent and previous search histories to re-rank search results. Therefore it is called query and time independent re-ranking strategy. However, given a user and her current query, no all the information in her user profile equally reflects the user's current search interest. To

overcome this shortcoming, we introduce a query and time dependent re-ranking strategy to distinguish the relevant search history from irrelevant search history given an input query in the next chapter.

Chapter 4

Personalized Re-ranking Using Query and Time Dependent Strategy

The metadata expressing topical categorizations of web pages in Open Directory Project (ODP) is manually entered and edited, and commonly considered as useful means for automated learning of user interests and user search behaviors. In Chapter 3, we discussed how these metadata can be exploited to achieve high quality personalized Web search. However, we note that the query and time independent re-ranking strategy used in Chapter 3 suffers from one potential problem. Regardless of current input queries, the strategy utilizes the whole learned user profile to re-rank search resutls. In this chapter, we propose a query and time dependent re-ranking strategy. We devise a query context window (QCW) based framework for query-centric *S*elective u*T*ilization of search history in personalized le*A*rning and re-*R*anking (STAR). Our framework includes a suite of algorithms for learning and encoding user's long-term and short-term search interests based on the hierarchical structure of topics provided by ODP and four QCW based re-ranking strategies using the hierarchical semantic similarity metrics.

The remainder of this chapter is organized as follows. In Section 4.1 we summary the current studies on personalized search and introduce our research motivation. The overview of our STAR framework is presented in Section 4.2. Then, we discuss building QCW based user profiles and designing re-rank strategies in Section 4.3 and Section 4.4 respectively. Experimental results and discussions will be given in Section 4.5. Finally, we conclude the chapter in Section 4.6.

4.1 Introduction

We categorize the research efforts on personalized search into three classes of strategies: 1) query modification or augmentation [15, 83], 2) link-based score personalization [37, 42, 60, 70, 73, 78], and 3) search result re-ranking [16, 24, 53, 54, 56, 59, 83, 92, 94]. A general process of re-ranking is to devise efficient mechanisms to re-order the search result ranking using the global importance by personalized ranking criteria. Such criteria are typically derived from the modeling of users' search behavior and interests. In the context of semantic Web, the utility of metadata describes the content and various other interesting properties of Web pages and relationships between them, which are useful for automated learning of user interests and re-ranking of search results.

In this chapter, we make use of the metadata in Open Directory Project (ODP) which expresses topical categorizations of web pages. The manually entered and edited ODP metadata are commonly considered as useful means for automated learning of user interests and user search behaviors. Based on ODP metadata, we propose a query context window (QCW) based framework for query-centric *S*elective u*T*ilization of search history in personalized le*A*rning and re-*R*anking (STAR). Our STAR framework consists of three design principles and a suite of algorithms for learning and encoding user's short-term and long-term search interests and re-ranking of search results through a careful combination of recent and previous search histories.

We argue that even though short-term interests based personalization using the most recent search histories may be effective at times [60, 82, 83], it is generally unstable and fails to capture the changing behavior of the users. Furthermore, most of existing long-term interests based personalization using the entire recent and previous search histories fails to distinguish the relevant search history from irrelevant search history [16, 72, 94], making it harder to be an effective measure alone for search personalization.

Bearing in mind of these observations, our *STAR* framework advocates three design principles for rank optimization. First, we devise a so-called query context window (QCW) model to capture the user's search behavior through a collection of her per-query based click-through data. Second, we develop a query-to-query similarity model to distinguish the relevant search memories of personalized search behavior from irrelevant ones in the QCW of each user, reducing the noises incurred by using either a recent fragment or the entire QCW. Third, we develop a fading memory based weight function to carefully combine the frequency of relevant search behavior (long term interests) with the most recent search behavior (short term interests). To show the effectiveness of our STAR framework in quality enhancement of personalized search, we propose length and depth based hierarchical semantic similarity metrics and compare the effectiveness

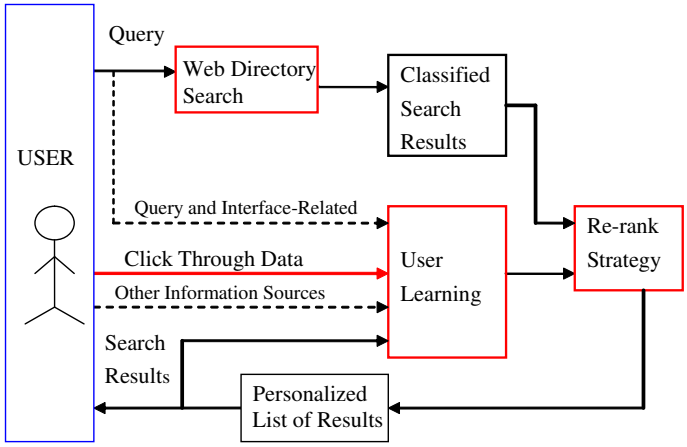

FIGURE 4.1: Overview of the STAR framework

of four re-ranking strategies: 1) naïve re-ranking that is query and time independent; 2) relevant search memory based re-ranking that is query dependent but time independent; 3) fading memory based re-ranking that is time dependent but query independent; and 4) hybrid re-ranking that is both query and time dependent.

Our experiments show that the hybrid re-ranking scheme can effectively combine the previous and recent memories through a smooth and gradually fading memory based weighting function. More importantly, our experimental results show that the proposed query-centric STAR framework for personalized re-ranking can effectively capture user-specific query-dependent personalization preference and significantly improve the accuracy of personalized search over the popular directory-based search methods (e.g., Google Directory search) and the general model of most existing re-ranking schemes of personalized search. We also show the variations in performance among queries with different search goals, especially when varying the weights on recent and previous click records of user search profiles.

4.2 The STAR Framework Overview

The goal of the STAR framework is to design a semantic rich user profile model to capture the query context and the search behavior of each user and intelligently utilize such user profiles to enhance the quality of personalized search, by effectively re-ranking of the search results returned from a general purpose search engine. The STAR framework, as shown in Figure 4.1 consists of three main components.

The first component is the Web directory search module that provides hierarchically structured access to high-quality content on the web. Open Directory Project (ODP) is

one of the largest efforts to manually annotate web pages, exporting all this metadata information in RDF format. Over 65,000 editors are working with keeping the directory reasonably up-to-date, and the ODP now indexes over 4 million web pages in the ODP catalog. Google uses ODP as basis for its Google Directory service and applies its patented PageRank technology on it, so we utilize the classified search results from Google Directory search.

The second component is the user-specific context aware learning of search behavior. We utilize the per-query based click-through data to capture query dependent context and search behavior and develop the query context window (QCW) model to encode such leaning process. By automatically generating QCW based user query profiles, the user learning module automatically captures the query dependent context of user search behavior. For example, our approach focuses on the user's visited search results (Web pages) which supply us with not only what kind of content a user is interested in (topics) but also how much the user is interested in them (click frequency).

The third component is the query and time dependent, hybrid re-ranking scheme that produces a new user-specific, query-dependent rank list for each user query through three step process. First, it selects the relevant click records from the entire QCW of a user through the query-to-query similarity analysis. Second, it combines the recent search memories with the previous search memories through applying a fading memory based weighting function over the selected QCW click records of a user. Finally, it employs hierarchical semantic similarity measures to compute the personalized ranking of search results.

In the subsequent sections we will focus on the technical detail of the user learning module and the re-rank module.

4.3 QCW Based User Learning

A simply way to learn user's interests is remembering her search history as a whole. However, not all the information in her search history is related to the current input query. We need to organize user's search history in a way that we can conveniently select query-dependent relevant information for rank optimization.

Our STAR framework addresses this problem and devises the query context window (QCW) to encode the user specific and query dependent search behavior. Given a user, her query context window consists of m query-dependent context click records, denoted as $u_1, u_2, \ldots, u_m$. Each click record in the QCW is composed of the submitted query, the topics associated with the click search results, the click frequency of each topic, and

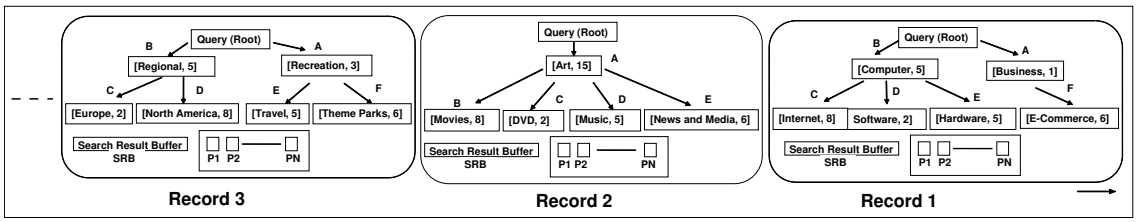

FIGURE 4.2: Query Context Window: click records are queued up in a chronological order (Record 1 is the head)

the returned search results of the given query. The topics are extracted from Google Directory, structured as a hierarchical tree, so that each click record has its own tree. This topic tree records the click behavior of a user on a specific query, which can tell us what kind of topics a user is interested in. The click frequency is an interest score representing how much the user is interested in this topic. The topic trees in all click records store user's interests. To help us choose relevant QCW click records given an input query, the search results $(P1, P2, \ldots, PN)$ responding to a past query are stored in the Search Result Buffer (SRB). The technical details of click record selection are in the next section. Moreover, we implement each QCW as a queue. The tail of the queue holds most recently requested queries, while its head holds the least recently requested queries. When a new query is submitted, the corresponding record is learned and added to the tail of the queue and QCW is updated accordingly. This queue keeps the chronological order of different click records, which can easily differ the recent and old search histories for re-ranking strategies.

Figure 4.2 shows an example of QCW with three context records, each corresponds to one query and its context encoding of the query dependent click-through data. For example, a user inputs a query "Disneyland" to Google Directory search engine, and then she clicks some search results. Record 3 in Figure 4.2 will store the input query "Disneyland" as a root node followed by the clicked topics. The search results are kept in the SRB. Node F is represented by the $[ThemeParks, 6]$ which means the user has clicked some search results associated with the topic "$ThemeParks$" six times in this search. In addition, for each topic, we store the top four depth of its full path in Google Directory in a record. For example, the node F is actually stored as the $[\backslash Recreation \backslash ThemeParks]$. We will discuss the effect of depth on retrieval effectiveness and efficiency in Section 4.5.

Our QCW user profile is learned from the user's click information. Although the click information is not as accurate as explicit relevance judgment in the traditional feedback, the user's choice does suggest a certain level of relevance. Therefore, we apply some selection techniques in addition to direct use of clickthrough data. Given a current input query, our re-rank module further extracts relevant contexts from the whole click

information which is relatively noisy if strictly speaking, but most of re-ranking based Web search personalization commonly used all available user click information as relevant contexts. We use click frequency to assign great weights on high frequent topics, which decrease the effect of low frequent topics on retrieval quality since these low frequent topics are higher possible to be noisy. We store the search results of past queries to choose relevant QCW click records given an input query, which filters the irrelevant click records from the whole click information. Our fading memory based weight function assigns more weights to more recent QCW click records and decreases weights to older QCW click records, which further improves the accurate utilization of user click information. Therefore, given an input query, using our re-rank module, we can extract clean and relevant contexts from the whole click information of a user, thus largely reducing the effect of the whole noisy click information of a user on retrieval quality.

4.4 Query-centric Re-ranking

Recall that our QCW based user profile contains the learned click records on a per-query basis. Each record encodes a query-dependent memory that stores the user's click behavior related to a query. The records in each QCW are queued up in a chronological order. In Figure 4.2 the head of the QCW (Record 1) holds the click record of the least recently issued query, which is regarded as a previous memory in the QCW relative to its tail. The tail (Record 3) keeps the click record of the most recently issued query (assuming the QCW has only three click records), which is considered as a recent memory in the QCW relative to its head.

The QCW based re-ranking module needs to address three key challenges. First, how to select relevant context records from the entire QCW given a user query. In other words, given a user and her current query, do all the context records in the QCW of the user equally reflect the user's current search interest? If not, which click records have higher probability in reflecting the current interests? Obviously if the user's current search interest is related to her short-term interests, the most recent context records are most likely to be useful. If the current user's search interest happens to be related also to her long-term interests, which means that the user has clicked the related topics often in the past, then both previous and recent memories are useful. Thus the second challenge is whether all the selected query-relevant context records play the same role in re-ranking the search results of the current query. If not, how to determine the weights of the selection of query-relevant context records? Finally, given the weighting function that combines previous and recent memories relevant to the current query, what is the most effective mechanism to compute the similarity of the current query results with

the selected QCW click records and how should we re-order the search results according to the similarity measures?

In the subsequent sections, we will describe our approach to address each of these three challenges in detail. Informally, we can view that long-term context generally holds long-term user's interests accumulated by experiences over a longer time period. Hence some interests may have appeared multiple times. The key challenge is that even if we can accurately obtain long-term context, would it possible that the entire long term QCW context be useful to every query of a user? For example, past searches like "animal" may be irrelevant to the current query "mouse" (computer input device). Such noise can overwhelm the related past searches. Surprisingly, most existing studies [16, 24, 59, 72, 83, 89, 94] did not distinguish between relevant and irrelevant parts of user context. On the other hand, short-term user's interests are unstable by nature. Interests in current hot topics could change quickly. In such cases, user context collected over a long period of time is unlikely to be useful, but the recent context, relatively speaking, can be much more useful in identifying the user's interests of the current query. A main challenge in exploiting short-term search history is the need to detect session boundaries [83], such that only those searches with high similarity are used, though properly determining session boundaries is non-trivial.

From the above discussion on state-of-art researches on Web search personalization, it is intuitive to think that we should first judge the current input query by a user is her long-term or short-term interests, and then re-ranking module will work on the whole previous and recent search memories for long-term interests or only on the recent memories for short-term interests. However, the accurate classification of two kinds of interests is not easy and the two kinds of interests have their own advantages and disadvantages. A suitable combination of the two kinds of interests would be useful for general searches.

Our key idea is not to explicitly classify the two kinds of interests, but to embed them into the previous and recent search memories of a user and then select and weight the relevant memories for a given query, no matter the current search is her short-term or long-term interests. We observe that the main difference between short-term and long-term user's interests can be captured by two factors. One is the time of the occurrence and the other is the frequency of the occurrence. The more recent the clicked topic is, the more likely it is to be the short-term interest. The more frequently the clicked topic appears in the QCW click records, the more likely it is to represent the long-term interest. We can say that short-term interests are mostly found in a user's recent click behaviors while long-term interests span the user's previous and recent click behaviors.

Based on these observations, we consider two factors critical in our STAR framework: (1) the query-to-query similarity which captures the relevant search memories of a user

TABLE 4.1: Symbols and their meanings

Symbol	Meaning
$\mathcal{U}$	the set of click records in QCW
u_i	the i_{th} element (click record) of $\mathcal{U}$
q^{u_i}	the query in the record u_i
$\mathcal{T}^{u_i}$	the set of topics in the record u_i
$t_j^{u_i}$	the topic j in the topic set $\mathcal{T}^{u_i}$
$\mathcal{C}^{u_i}$	the set of the interest scores in the record u_i
$c_j^{u_i}$	the interest score of the topic j in the record u_i
$\mathcal{SRB}^{u_i}$	the set of search results stored in the SRB of the record u_i
q_{in}	the input query
$\mathcal{P}^{q_{in}}$	the set of search results given the input query q_{in}
$p_k^{q_{in}}$	the k_{th} search result given the input query q_{in}
hf	the half fading parameter
h	the depth of the subsumer of two topics
l	the shortest path length between two topics
M	the maximum depth of topic directory processed by our framework

including user's long-term and short-term interests, and (2) a fading memory based function which assigns greater weights on the recent search memories than the previous ones since the most recent ones reflect the short term interests of a user. In general, the more similar the QCW encoded context record is to the current input query, and the more recently the QCW click record appears, the higher probability that the QCW click record is representative in terms of the user interest of the current search.

In the remaining of the chapter, we use calligraphic upper-case alphabets to represent sets. The elements of a set are denoted by lower case alphabets. For example, u_i is an element (click record) of $\mathcal{U}$ (the set of click records in QCW). $|\mathcal{U}|$ is the cardinality of the set $\mathcal{U}$. Symbols used in this chapter and their meanings are listed in Table 4.1.

4.4.1 Selecting Relevant Click Records by Query-to-Query Similarity

Given a new input query, we first select the relevant QCW click records where the encoded queries are similar to the current input query by using a query-to-query similarity measure. Estimating the similarity (relatedness) between queries has a long history in traditional Information Retrieval and it is still hot and active in various topics of Web Information Retrieval [75, 97, 108]. One of the lessons learned in the Information Retrieval area is that there are various similarity and specificity measures as well as various ways of combining them. Up to now it has not been possible to prove that any of these measures outperforms all others in a large set of experiments [108].

In our case, it is intuitive to use the term overlap between two queries as a similarity measure (e.g., previous queries having common terms with the input query are naturally recommended as alternatives). However, only a couple of keywords are used in defining Web queries [97]. These short Web query are deficient in accurately representing user's needs. It is possible that queries may be identical or phrased differently with different terms but for the same information needs. Consider the example of two queries, "IRS (Internal Revenue Service) form" and "file taxes online". Although they have no terms in common, both of them concern the application of filing taxes. The similarity between the two queries can be induced from the overlap of the two lists of search results (URLs) returned. Clearly, the query-result-vectors present a better similarity metric than query term-vectors [75].

As thus, using the query result URL to encode each search result, we formally define the query-to-query similarity measure as follows:

$$Q(q^{u_i}, q_{in}) = \frac{|\mathcal{SRB}^{u_i} \cap \mathcal{P}^{q_{in}}|}{|\mathcal{SRB}^{u_i} \cup \mathcal{P}^{q_{in}}|}. \tag{4.1}$$

Given the past query q^{u_i} in the click record u_i of a QCW-based user profile and the current input query q_{in}, we can get the URL set of search results of q^{u_i} from the search result buffer $\mathcal{SRB}^{u_i}$ of the click record u_i, and the URL set of search results $\mathcal{P}^{q_{in}}$ of q_{in} from the current search. The similarity between the two queries is estimated to the fraction of the intersection of the two URL sets (i.e., $\mathcal{SRB}^{u_i}$ and $\mathcal{P}^{q_{in}}$). Our query-to-query similarity measure states that the more URLs two queries have in common in their result sets, the more similar they are. The value of the defined similarity between two queries lies in the range [0, 1]: 1 if they have exactly the same URLs, and 0 if they have no URLs in common. In our experiments, URL similarity is measured by their host name.

Though our URL-based query to query similarity measure is simple and intuitive, our experiments show that it can effectively extract relevant click records from QCW. We would like to note that our STAR framework can easily incorporate other similarity and specificity measures. Due to the space constraint and the fact that this chapter focuses on combining query-to-query similarity with long and short term memory functions to improve the re-ranking of search results, we omit the further discussion on more complex query to query similarity measures.

4.4.2 Weighing Relevant Click Records Using Fading Memory Model

Equation 4.1 answers the first question that is how to select relevant context records from the entire QCW given a user query, which can select semantic similar queries,

and then the user context with these queries in QCW is relevant to the current user's information need phrased as the input query. Here, we address the second challenge that is whether all the selected query-relevant context records play the same role in search results re-ranking.

These selected click records are the collection of user's previous and recent search behaviors which reflect her interests. We assume that the user's interests will gradually decay as time goes on, so we assign more weights to more recent QCW click records and decreasing weights to older QCW click records to further improve the accuracy of the personalized search using a fading memory based weight function, defined as follows:

$$F(u_i) = e^{-\frac{log2}{hf \cdot |\mathcal{U}|} \cdot (|\mathcal{U}| - i)}, \tag{4.2}$$

where hf is a half fading parameter. In our experiments, hf is set in the range [0.1, 1]. After the click record u_i is selected as relevant according to the similarity between its q^{u_i} and the current input query q_{in}, its effect on the quality of personalized search (i.e., $F(u_i)$) depends on its temporal order. For example, if the click record u_i is located in the middle of the whole QCW (i.e., in the center of the oldest memory and the most recent memory, namely i=$|\mathcal{U}|/2$), its effect will be reduced by 1/2 when hf=0.5. With increasing the value of hf, the rate of fading becomes slow and the weights on previous memories increase. In our STAR framework, this fading memory function is a key metric to unify the user's long-term and short-term interests encoded in the QCW click records by assigning different weights to these click records appearing in different temporal order.

4.4.3 Capturing Search Interests Using Topic Similarity Measure

We have defined the query-to-query similarity and the fading memory based weight function for selecting QCW click records related to the current search interest of a user. After the relevant QCW click records and their weights are determined, the topics in these QCW click records are reflecting the user's current search interests. Now we can devise a re-ranking mechanism to re-order the search results by putting those that are more similar to the selected topics closer to the top of the final re-ordered rank list.

In other words, given one of search results of the input query q_{in}, (e.g., $p_k^{q_{in}}$, the kth search result) and the set of weighted relevant QCW click records, we calculate the similarity between them. The higher similarity score $p_k^{q_{in}}$ is getting in comparison with the results of historical queries of the same user, the higher position it will be placed in the final ranked result list. In our STAR framework, the topics in the relevant QCW click records are structured in a semantic concept hierarchy as shown in Figure 4.2.

Hierarchical similarity measures can be used to assess the similarity between the related topics and the search results of the given query.

There are a set of topics $\mathcal{T}^{u_i}$ in the click record u_i and each search result is classified to a topic, so we first compute the hierarchical similarity score between each topic $t_j^{u_i}$ in $\mathcal{T}^{u_i}$ and the topic of $p_k^{q_{in}}$, and then combine all the scores of the topics in $\mathcal{T}^{u_i}$. We exploit the structural similarity among the related topics by considering the length-depth hybrid hierarchical similarity measures (i.e., Equation 4.7-4.8) [57] since the length based (i.e., Equation 4.3-4.4) and the depth based (i.e., Equation 4.5-4.6) have their own shortcomings, as shown in our analysis below.

Let h be the depth of the subsumer (the deepest node common to two nodes), l be the shortest path length between two topics, and M be the maximum depth of topic directory possessed by a QCW click record. We will experimentally discuss the setting of M in Section 4.5.

1) Length-based Topic Similarity Measure

The length-based measure is intuitive and considers the shortest path length between two topics (nodes) alone. Its linear and exponential form are defined as:

$$L1 \equiv HS(t_j^{u_i}, p_k^{q_{in}}) = 2 \cdot M - l\,, \tag{4.3}$$

$$L2 \equiv HS(t_j^{u_i}, p_k^{q_{in}}) = e^{-0.25 \cdot l}\,. \tag{4.4}$$

Clearly, Equation 4.3 is a linear function of the shortest path length between two topics (nodes) and Equation 4.4 measures the same using a nonlinear function. Consider the example in Figure 4.2, given node A and node B in Record 1 of Figure 4.2, the length between them is 2, so their similarity values computed by $L1$ and $L2$ are 6 and 0.6065 respectively. The main drawback of the naïve length-based similarity is that it overlooks the depth of the subsumer. For example, in Record 1 of Figure 4.2, the length between node C and node D is also 2, which produces the same similarity value as that between node A and node B. However, the similarity between node C and node D should be different from that between node A and node B since the subsumer of node C and node D (i.e., node B) is in different levels of the hierarchy from the subsumer of node A and B (i.e., *Root*).

2) Depth-based Topic Similarity Measure

An obvious alternative topic similarity measure is to consider the depth of subsumer information instead. The following two depth based equations use linear and nonlinear functions respectively to measure the topic similarity of the current user search query

with her previous queries:

$$D1 \equiv HS(t_j^{u_i}, p_k^{q_{in}}) = 0.05 \cdot (2 \cdot M - l) + h\,, \tag{4.5}$$

$$D2 \equiv HS(t_j^{u_i}, p_k^{q_{in}}) = \frac{e^{0.15 \cdot h} - e^{-0.15 \cdot h}}{e^{0.15 \cdot h} + e^{-0.15 \cdot h}}\,. \tag{4.6}$$

Although Equation 4.5 uses the length in its definition, it gives relatively much heavier weight on the depth (1 vs. 0.05). Therefore, we classify it into depth-based metrics. Equation 4.6 is the transformation of the depth of the subsumer through a nonlinear function. Using node A and node B in Record 1 as an example again, the depth of their subsumer is 1 (i.e., $Query(Root)$ node), so the similarity values computed by $D1$ and $D2$ are 1.3 and 0.1489 respectively. It is easy to understand that using the depth information alone is also not optimal. For example, the subsumer of node B and node A is the same as that of node B and node F, but the length between node B and node A (i.e., 2) is shorter than that between node B and node F (i.e., 3). Consequently, we consider the combination of length and depth used in the hierarchical semantic similarity.

3) Length & Depth based Topic Similarity Measure

Motivated by the strength and weakness of length and depth based approaches, we use a careful combination of depth and length based similarity measure defined as follows:

$$C1 \equiv HS(t_j^{u_i}, p_k^{q_{in}}) = \frac{2 \cdot h}{l + 2 \cdot h}\,, \tag{4.7}$$

$$C2 \equiv HS(t_j^{u_i}, p_k^{q_{in}}) = e^{-0.2 \cdot l} \cdot \frac{e^{0.6 \cdot h} - e^{-0.6 \cdot h}}{e^{0.6 \cdot h} + e^{-0.6 \cdot h}}\,. \tag{4.8}$$

Equation 4.7 is a simple linear transformation function of the length and the depth, while Equation 4.8 transfers the length and the depth by a nonlinear function and then combines them by multiplication. The similarity values between node A and node B in Record 1 calculated by $C1$ and $C2$ are 0.36 and 0.5 respectively. We expect that the combination can work well for common cases since using depth or length alone has its shortcomings in some conditions. The parameters used in these equations weighs the depth and length. In this chapter, we do not address how to get the best weights which is well discussed in [57].

The above six equations define the similarity between two topics (nodes), which we use, in combination with the query to query similarity measure, to capture the similarity of a user's current search behavior with her previous search behavior. A QCW click record may record more than one topic depending a user's click behavior. We further define the similarity between a QCW click record u_i and a search result $p_k^{q_{in}}$ as:

$$S(u_i, p_k^{q_{in}}) = \frac{1}{|\mathcal{T}^{u_i}|} \sum_{t_j^{u_i} \in \mathcal{T}^{u_i}} \frac{HS(t_j^{u_i}.p_k^{q_{in}}) \cdot c_j^{u_i}}{\sum_{c_j^{u_i} \in \mathcal{C}^{u_i}} c_j^{u_i}}\,, \tag{4.9}$$

where each topic $t_j^{u_i}$ in $\mathcal{T}^{u_i}$ is weighted by its corresponding $c_j^{u_i}$ representing the interest score of the topic j in u_i. The larger an interest score is, the more interested the user is in one topic.

In this chapter, we use click frequency as the interest score. We obtain a normalized version by dividing this score by the sum of the interest scores of all the topics in u_i. Then we sum all the normalized weighted hierarchical similarity scores of the topics in u_i with a search result. Moreover, the number of topics stored in each click record depends on user's click behavior. The more clicked topics in a click record, the click record may gain larger similarity scores. Due to the collective strength of these topics, although each clicked topic in it may have a relatively small score, the sum of these scores will effect the re-ranking quality. We further normalize the sum of hierarchical similarity scores through dividing it by the number of topics stored in a click record $|\mathcal{T}^{u_i}|$. We want to mention although our evaluation utilizes the search results which have already been classified to their corresponding topics in Google Directory search engine, our approach can also be applied in other search engines by adding a Web page classifier.

4.4.4 QCW Based Re-ranking

We have discussed how to encode the search behavior of a user in QCW (Section 4.3), how to select relevant QCW click records for a given user query, how to determine the weights of the selection of query-relevant QCW click records, and how to compute hierarchical semantic similarity between a QCW click record and a current query in Section 4.4. In this section we will describe how to use all the selected relevant QCW click records of the given user to re-order the search results of her current query.

To provide a better understanding of the effects of different factors on the quality of our hybrid re-ranking optimization, we consider the following four re-ranking strategies. The first one is the naïve strategy that assumes the equal role for all QCW click records in evaluating the user's interest of the current query. The second scheme is to highlight the role of the relevant QCW click records in the re-ranking algorithm. The third scheme is to focus the time factor by a fading memory weighting function, which assign more weights to more recent QCW click records. The fourth scheme re-ranks the current search results by carefully combining previous and current memories through the query-to-query similarity and the fading memory weighting function.

1) Strategy 1 – Query and time independent scheme

$$S_1(\mathcal{U}, p_k^{q_{in}}) = \frac{1}{|\mathcal{U}|} \sum_{u_i \in \mathcal{U}} S(u_i, p_k^{q_{in}}) \,. \tag{4.10}$$

"Strategy 1" is query and time independent, a naïve strategy, which defines an equal weighting strategy. Click records of different past queries are assigned equal weights regardless of the current input query. The similarity scores of past queries with a search result $p_k^{q_{in}}$ are summed together and divided by the number of click records ($|\mathcal{U}|$) in $\mathcal{U}$.

There is no selection of relevant click records and no temporal order based weighting in "Strategy 1", which means all the past histories (click records) are related to a user's current query. This simple weighting schemes suffers from the problems that it produces a global but weak description of user's current search interests. As we discussed in Section 4.1, the entire QCW includes noisy memories unrelated to the current query and only those that are related to the current search interests are important. We should therefore assign much weight on them, and ignore other noisy memories of QCW. Most of re-ranking based Web search personalization methods in the literature [16, 24, 59, 72, 83, 89, 94] and our previous works [53, 54, 56] have commonly used all available user context to get some improvement. "Strategy 1" can represent the general idea of these methods, compared with the following three strategies.

2)Strategy 2 – Query dependent scheme

$$S_2(\mathcal{U}, p_k^{q_{in}}) = \frac{1}{|\mathcal{U}|} \sum_{u_i \in \mathcal{U}} Q(q^{u_i}, q_{in}) \cdot S(u_i, p_k^{q_{in}}) \,. \tag{4.11}$$

We define the "Strategy 2" as a query dependent and time independent strategy, which is selective about which click records of QCW to use according to the current query q_{in} by using the query-to-query similarity $Q(q^{u_i}, q_{in})$ to weight these click records. Tan et al. [92] did preliminary discussion on query-dependent selection of user profile. However, their work is in the context of only exploiting long-term search histories of users and ignores the changes of user's interests with time. For example, if a Web user used the query "Python" to get information on snake ago, and now she is interested in Python programming language and search related programming skills on the Web. When she inputs "Python" on a search engine again, it is reasonable to think that the recent search histories on Python in the field of computer science are more important than the previously clicked Web pages on snakes.

3)Strategy 3 – Time dependent scheme

$$S_3(\mathcal{U}, p_k^{q_{in}}) = \frac{1}{|\mathcal{U}|} \sum_{u_i \in \mathcal{U}} F(u_i) \cdot S(u_i, p_k^{q_{in}}) \,. \tag{4.12}$$

"Strategy 3" strengthens recent memories and weakens the effect of previous memories by applying the fading function $F(u_i)$ to each QCW click record without the selection of relevant contexts in terms of the input query like "Strategy 2". If hf is set to a very small value, the previous memories cannot have an influential effect on re-ranking. Then

for simplicity we can think that in this case the quality of "Strategy 3" is largely based on the recent memories and ignores the previous memories even if these old memories are related to the current query.

Researches [60, 82, 83] emphasize that the most recent search is most directly close to the user's current information need, which can be regarded as a special case where hf is close to zero in "Strategy 3". The retrieval quality improved by their approaches are heavily relied on accurately detecting session boundaries, such that only those searches within the session are used as relevant search histories. Properly finding session boundaries is non-trivial, so they determined the session boundary by manual or using 30 minutes, a well-known threshold. Based on these discussions, we think the combination of query-dependent and time-dependent strategy would be more effective in a general Web search.

4) Strategy 4 – Query and time dependent scheme

$$S_4(\mathcal{U}, p_k^{q_{in}}) = \frac{1}{|\mathcal{U}|} \sum_{u_i \in \mathcal{U}} F(u_i) \cdot Q(q^{u_i}, q_{in}) \cdot S(u_i, p_k^{q_{in}}) \,. \tag{4.13}$$

"Strategy 4" is both query and time dependent, a hybrid strategy. As we know, users have their own characteristics of search behavior. If a user always likes phrasing general queries which cover a number of various topics, her whole search histories would be useful like "Strategy 1". If a user major in some field, e.g., Information Retrieval, likes to frequently search the latest technical reports , this interest can be captured and then the search histories on IR can be extracted from her whole histories using "Strategy 2". If all the searches of a user are related to different information needs and there is no relatedness between these information needs, we cannot learn her long-term interests which should be consistent and accumulated by experiences over a long time period. In this case, recent-memory based strategy like the time dependent "Strategy 3" is effective. If the query input by a user may have no relevance with her search history, the current general search engines are doing a good job and then no personalization is needed. To handle the most general case where we have many kinds of Web users and users will show different search behaviors, "Strategy 4" is designed to select relevant click records $Q(q^{u_i}, q_{in})$ and assign greater weights to the more recent click records $F(u_i)$.

Given one of the four strategies, a new relevant score will be calculated for each of search results. We output the list of the search results in order of their assigned scores. We use a concrete example to further discern the four re-ranking strategies. Recall Figure 4.2, given the query "Disneyland", we assume that the relevant click records are Record 2 and Record 3 in Figure 4.2 which includes topics like "Theme Parks", "Travel", "Music" and so on. This assumption means that the query-to-query similarity scores (Equation 4.1) of the two click records are not zero. "Strategy 1" will use all the click records to re-rank

search results, so that if there are unrelated Web pages about "Computer" or "Business" in the search results, they will be considered related to the input query, thus lowering the retrieval accuracy. "Strategy 2" with the help of Equation 4.1 can identify Record 2 and Record 3 as relevant click records, thus expelling the noisy Record 1 from the re-ranking scheme. Although "Strategy 3" gives less weights to Record 1 than Record 2 and Record 3, Record 1 is not the relevant click record. Thus, this strategy will be interfered by Record 1 and cannot benefit much from the relevant Record 2 and Record 3. Based on the selected relevant click records, "Strategy 4" can not only select the relevant Record 2 and Record 3, but also assign greater weight on Record 3 than Record 2 by using Equation 4.2 because Record 3 is more recently added into QCW than Record 2. In the following experiments, we will evaluate the effectiveness of the four re-rank strategies utilized in our STAR framework.

4.5 Experiments

4.5.1 Experiment Setup

The goal of this chapter is to achieve a personalized ranking by scoring the similarity between a user profile and the returned search results. Instead of creating our own Web search engine, we retrieve results from Google Directory search engine and use them as a baseline in the following evaluation.

In speaking of user-centric Web search behaviors, there are three commonly accepted classes of user search scenarios. An extreme scenario is the case where a user surfs the Web without a specific goal, and thus her click data is completely random. This is an extreme case which, we consider as irrelevant since her search history is totally irrelevant to her current information need. As a matter of fact, this scenario does not existing in real life. Even for kids or grandma their surfing has some patterns with some objective. The second scenario is when user search behavior follows some regulated behavior with a clear search goal. This is typically the case for professionals, researchers and specialists who use the Web for their research. The third and more common scenario is the case when the users are not completely focused with search goal but over time their searches still follow some patterns, such as kids most interested in kid friendly Web contents. Grand parents with cooking interests may visit more frequently the cooking Web contents. We believe that the latter two kinds of search scenarios (regulated, common) represent most users' search behaviors on the Web.

Our work focuses on the query-centric re-ranking of search results. Informational queries (IQ) [79] are such queries where the user does not have a special page in mind and intends

to find out Web pages related to a topic. Thus, we further classified the goal of IQ into three categories: new IQ, semi-new IQ, and repeated IQ. A query is a new IQ if a user has never searched such a topic before. It means that we cannot get the relevant search histories from this user's QCW profile. A semi-new IQ is an IQ that has similar topical contents with some past queries. A repeated IQ refers to the type of queries that have been asked previously and had already obtained the desired information by the user. The user is simply, repeating the search again for updated information.

It is interesting to note that there is some intrinsic connectivity between these two types of search behavior classifications. If a user (random) never surfs the web with a specific goal, then new IQ can represent such search behavior since the user's search history cannot help much in her future search. However, if a user (regulated) is a researcher or specialist, she only has a large portion of highly specialized expert queries and thus a small range of topic categories. Repeated IQs can represent this scenario well, since in simple case we assume that her previous query may be repeatedly issued for several times. The third and most common search scenario is the case where users, though do not have clear goal in each search, but we see some similar interests over a long period of time. Thus semi-new IQs are well suited to represent this type of queries. In the rest of this section, we focus our experiments on evaluating the performance of the semi-new IQs and repeated IQs since we want to show how our STAR framework utilizes the previous relevant search memories to enhance the current search. One of our future work is to study collaborative information retrieval techniques to improve the effectiveness of processing new IQs.

4.5.1.1 Real Data Set

It is well known that the evaluation of automated learning of user interests is challenging because currently there are no suitable query log data sets made available as a public benchmark. For this experimental study, we used a real data set and a synthetic dataset. The real data set was collected over a ten-day period (From October 23rd, 2006, to November 1st, 2006). Twelve users are invited to search through our framework and judge whether the clicked results are relevant or not. They are graduate students (5 females and 7 males). Users were asked to input search queries related to their professional knowledge in the first four days, and search queries related to their hobbies in the next three days. Then, in the last three days, each user is requested to repeat some searches with the queries entered in the previous days. We got a log of about 300 queries averaging 25 queries per subject and about 1200 records of the pages the users clicked in total. The size of this real data set is relatively small because the click data collection and users' judgments are labor intensive.

4.5.1.2 Synthetic Data Set

We create a synthetic data set to perform parameter analysis of the proposed approach without putting much burden on users. However, generating a suitable synthetic data set is not an easy task either. Generally speaking, we need three data sources to generate our synthetic data set: queries, users' click behavior, and relevance judgment.

First, we need a collection of queries that reflect users' information needs. Moreover, there should be either a chain of queries for the similar information need, or a reformation of some previous query, so that we can evaluate the effectiveness of our re-rank strategies in utilizing the previous relevant memories to enhance the current search. We use the original queries collected by Shen et al. [82] [1] as the seed set since those queries in this seed collection, which are semantically related, are labelled by a same topic number (30 topics and 146 queries in total). We use the topic number to represent one specific requirement of a user, and the queries within this topic are submitted in successive searches, to simulate user's querying behavior.

The second dataset we need to create is the user click data, which should reflect Web users' click behavior. Accurately learning user's search behavior is still a difficult problem at present and there is no commonly accepted general user click model for us to use. The user model proposed in [73] is based on the assumption that the user's clicks are solely determined by the rankings of search results. This model, however, is somewhat over-weighted the search engine's ranking results in the user click behavior. We argue that such a user click model is inadequate in personalized re-ranking research that weights more on the personal context in search efficiency than general ranking of a search engine.

We argue that the user click model should take into account of both general ranking from a search engine and the personal search history and context information. In building such a user click dataset, we retrieved the top 50 search results of each queries from Google Directory Search, and the number of total search results is 6536. For each topic number (including several queries), we also obtained the topical categories of search results. The search results associated with a category that has the maximal occurrence frequency among all the categories, will be set as clicked ones. Five users are generated according to different click sizes. The lower bounds of their click sizes are 5, 15, 25, 35, 45, respectively. If the number of search results, whose category has the maximal occurrence frequency, is smaller than the lower bound, then the category with the second maximal occurrence frequency will be selected to generate click-through data. This process continues, until the click size reaches the lower bound. This user click model is not only simple but also reasonable and close to reality for our evaluation.

[1] http://sifaka.cs.uiuc.edu/ucair/QCHistory.zip

The third dataset that we need to generate is user's judgments for each search result. Deciding the relevant Web pages of a topic number follows the same idea as generating click data, which means we utilize click decisions as relevance judgments to evaluate the search accuracy. Although the click-through data in real world are more noisy, especially with random surfers with no goal included, and the actual performance score of our evaluation will be somewhat lower by the noisy information, our use of the synthetic data set in the experiments is primarily on studying the effect of different settings of parameters and different scenarios of queries on re-ranking quality. Thus our evaluation concerns more with the trend of the change in user behavior rather than the actual values.

4.5.2 Evaluation Measure

Precision is a standard measure in the field of information retrieval. We calculate a normalized precision because we test 30 queries. First, the average precision of a single query is defined as:

$$AvgP = \frac{\sum_{k=1}^{N} P@k \cdot sp_k^{q_{in}}}{\sum_{p_k^{q_{in}} \in \mathcal{P}^{q_{in}}} sp_k^{q_{in}}}. \tag{4.14}$$

$sp_k^{q_{in}}$ is the user's judgment on the k_{th} search result of the query q_{in} and it has two values, 0 for "irrelevant" and 1 for "relevant". $P@N$ is the number of relevant results considering only the top N results returned by the system (e.g., N=15). For a single query, average precision is defined as the average of the $P@k$ values for all relevant documents (k=1, $\cdots$, N). We divide the sum of the $AvgP$ values of all the queries by the number of testing queries (i.e., 30), which represents the *normalized average precision* and is used as one of our evaluation measures.

4.5.3 Evaluation on Real Data Set

We report the experimental results on the real data set. The queries in the last three days are regarded as repeated IQs. The first seven-day click-through data is divided into two parts (odd-day and even-day) as semi-new informational searches. One is for setting up the QCW user profile and the other is for re-ranking search results based on the learned user profile, and then the two parts are exchanged to run the evaluation once again.

In Figure 4.3 and Table 4.2, we summarize the performance of the proposed four re-rank strategies according to different hierarchical semantic measures. "Normalized

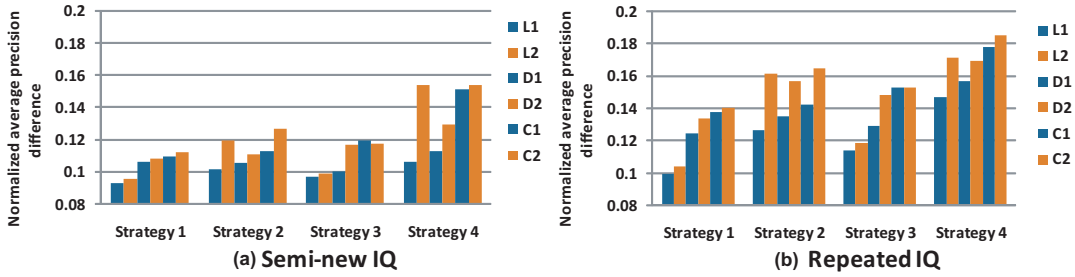

FIGURE 4.3: The improvement difference values of our strategies on real data set

TABLE 4.2: The improvement percentage of our strategy on real data set

Semi-new IQ (%)					Repeated IQ (%)				
Measure	Strategy1	Strategy2	Strategy3	Strategy4	Measure	Strategy1	Strategy2	Strategy3	Strategy4
L1	21.17	23.10	22.04	**24.11**	L1	40.27	51.43	46.18	**59.67**
L2	21.72	27.07	22.50	**35.00**	L2	42.36	65.43	47.99	**69.53**
D1	24.14	23.93	22.83	**25.60**	D1	50.42	54.85	52.41	**63.54**
D2	24.61	25.61	26.49	**29.34**	D2	54.19	63.67	60.20	**68.54**
C1	24.92	25.71	**27.13**	**34.35**	C1	55.95	57.72	61.92	**72.16**
C2	**25.50**	**28.78**	**26.64**	**34.88**	C2	**56.93**	**66.71**	**61.92**	**75.00**

average precision difference" means the difference value between our strategy and the baseline and "Normalized average precision % " represents the improvement percentage of our strategy over the baseline. The normalized average precision score of the baseline is around 0.55. The experimental results show that the proposed user-context aware re-rank strategies are more effective than the baseline. "Strategy 1", representing the general idea of most existing personalized re-ranking schemes, is inferior to other three strategies. Among the proposed four re-rank strategies, the "Strategy 4" broadly shows the best performance. The "Strategy 2" with selective utilization of user profiles, averagely produces better results than the "Strategy 1" and "Strategy 3". In Figure 4.3 and Table 4.2 the improvement of repeated IQs in is more obvious than those of semi-new IQs. The larger improvement of repeated IQs shows that our re-rank strategies can effectively retrieve the Web pages previously clicked by users since these queries have been submitted before and user's click behavior has been stored in our QCW.

Moreover, in Figure 4.3 and in Table 4.2 we observed that the similarity measures using nonlinear transformation function (i.e., $L2$, $D2$, and $C2$ shown in orange columns) generally produce better performance that the similarity measures using linear transformation (i.e., $L1$, $D1$, and $C1$ shown in blue columns). $C2$, the combination of length and depth with nonlinear transformation, generates the highest improvement among all the six measures. Length-based nolinear measure $L2$ largely increases its performance in "Strategy 2" and "Strategy 4". The two strategies using Equation 4.1 select relevant click records given an input query and filter some irrelevant records. Therefore, using length information alone can gain comparable performance with $C2$. In a word, "Strategy 4" with $C2$ produces the largest improvement, e.g., its improvements over baseline

TABLE 4.3: Effect of M on retrieval quality

M	1	2	3	4	5	6	all
NAP	0.39	0.57	0.74	0.86	0.88	0.88	0.88
Run time(ms)	1175	1175	1176	1178	1186	1189	1199

are 34.88% and 75% for semi-new IQs and repeated IQs respectively.

4.5.4 Evaluation on Synthetic Data Set

The evaluation on synthetic data set focuses on the effectiveness of different settings of parameters and different kinds of queries on retrieval and re-ranking quality. We use the last (most recent) queries in each topic as testing queries. The number of testing queries is 30. When the testing queries are regarded as repeated IQs, the whole click-through data is used to set up QCW (i.e., 146 click records). When they are treated as semi-new IQs, the click-through data of these testing queries is extracted from the whole data set. The remaining data is used as training data to build QCW (i.e., 146-30=116 click records).

4.5.4.1 Effect of the Depth of Topic Directory

In our QCW user profile, the topics in each record extracted from Google Directory are used to re-rank search results. We first do a preliminary analysis on the effect of the depth of the full path of a topic on retrieval efficiency and effectiveness (i.e., M in Section 4.4). In this evaluation click size is 25 and hierarchical semantic measure is "C2". We use "Strategy 4" for repeated IQ and set hf=0.5 and the size of SRB be 50 (we will discuss these parameters soon). Table 4.3 gives the retrieval efficiency and effectiveness of the depth of topic tree, where "NAP" represents normalized average precision. It shows that the deeper the topic directory we process, the larger improvement is reached while the run time increases. When the depth of topic tree is changed from one to four, the improvement of retrieval quality is increasing greatly. However, when the depth is increased continually from four to full path, the amount of quality improvement changes relatively slowly. In addition, the run time of using the top five depth of directory path becomes much longer that that of using the depth smaller than four. Due to this observation and the large size of the whole Google Directory, the top four topic directory path is encoded in our QCW user profile (i.e., M=4).

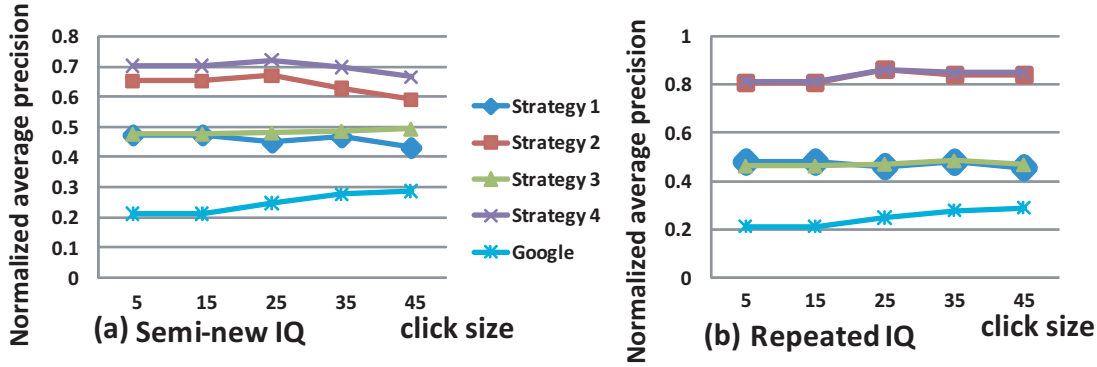

FIGURE 4.4: Effect of re-ranking strategies on effectiveness

4.5.4.2 Effect of Four Re-ranking Strategies

One of the key components in our framework is the selection of relevant click records from QCW using our proposed Equation 4.1 which is affected by the number of search results. The search results are stored in the SRB of each click record of QCW, so we discuss the effect of the SRB size on retrieval quality. We analyze "Strategy 4" with hierarchical semantic measure "C2" when click size=25 and hf=0.5. The performance of repeated IQs becomes a little better from 0.861 to 0.864 when the number of top search results in SRB changes from 10 to 50 with step 10. Equation 4.1 relying on the number of common URLs of search results of two queries, is computationally easy. Although the performance improvement is not significant, in the following experiments, we still set the size of SRB to be 50.

We now study the performance of the proposed four re-ranking strategies as shown in Figure 4.4. $C2$ hierarchical similarity measure is used and the value of hf is 0.5. Instead of using the QCW user profile as a whole, the hybrid re-ranking scheme "Strategy 4" shows the best performance in all cases. The "Strategy 2", winning the second place, assigns greater weights on the selected click records, thus reducing the effect of the noisy click records on retrieval quality. It, however, ignores the factor that user's interests will gradually decay with time goes on. Simply applying time factor to weigh all click records without eliminating irrelevant click records like "Strategy 3" is not an optimal scheme as well. "Strategy 2, 3, 4" generally produce higher retrieval quality than "Strategy 1" (a general model representing the common ideas of re-ranking based personalized search in the literature). Moreover, all the four strategies outperform Google Directory Search in terms of precision.

Furthermore, one interesting observation is the variations in performance between semi-new and repeated IQs, especially when we compare "Strategy 4" and "Strategy 2".

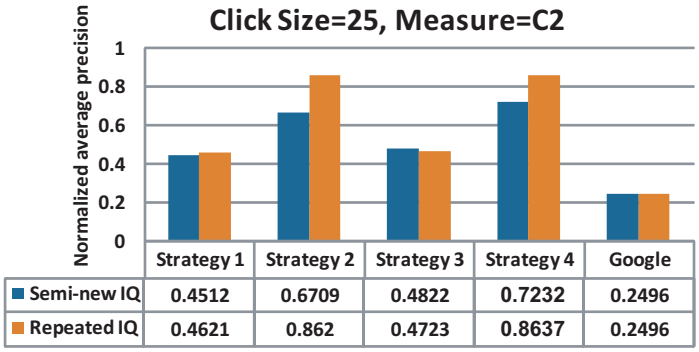

FIGURE 4.5: Results of semi-new and repeated IQs

The difference between the two strategies is whether the fading memory based weighting function is utilized. For example, for semi-new IQs in Figure 4.4(a), "Strategy 4" generally produces higher precision scores than "Strategy 2"; while for repeated IQs in Figure 4.4(b) "Strategy 2" competes with "Strategy 4". This observation tells us that for semi-new IQs, the user's current search need can be mostly inferred from the short-term memories (more recent click records). As a result, the short term memories have more effect on retrieval quality than the long term ones, so that the time factor becomes important in the re-ranking scheme. On the other hand, repeated IQs are more likely to reflect long term user's interests since the user wants to refind some information by submitting a same query again. The re-ranking scheme should add the weights for long term memories. To understand it from a quantitative view, we list the normalized average precision scores in Figure 4.5. For example, for repeated IQs "Strategy 3" cannot improve "Strategy 1" much and their precision scores are 0.4723 and 0.4621, respectively. The "Strategy 4" produces higher precision scores than the "Strategy 1" for both two kinds of IQs.

4.5.4.3 Effect of Topic Similarity Measures

In this part, we show our re-ranking retrieval quality with the effect of the six hierarchical semantic similarity measures to further discuss the selective utilization of search histories. Due to page limit and the similar results between repeated and semi-new IQs, the results of semi-new IQs averaged over the five users are illustrated in Figure 4.6. The similarity measures using nonlinear transformation function (i.e., $L2$, $D2$, and $C2$ shown in orange columns) still produce better performance that the similarity measures using linear transformation (i.e., $L1$, $D1$, and $C1$ shown in blue columns). For example, $L1$ uses the same structural information (i.e., length) as $L2$, but it produces the lowest retrieval precision among the six measures. Moreover, for the depth and length combined measure $C2$ distinctly shows higher retrieval precision than other five measures.

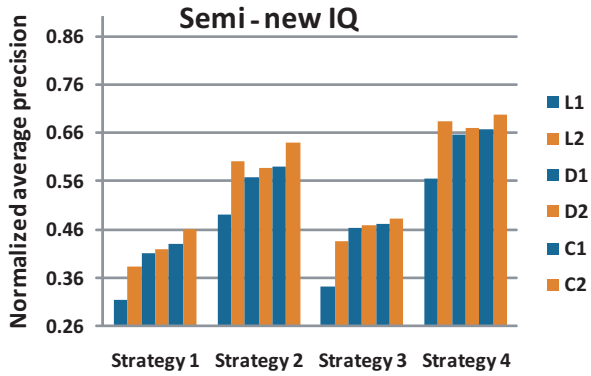

FIGURE 4.6: Effect of similarity measures on effectiveness

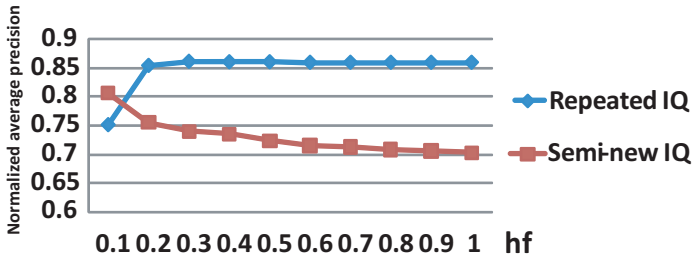

FIGURE 4.7: Effect of the parameter hf on effectiveness

However, if "Strategy 2" or "Strategy 4" is applied in our framework, $L2$, a length-based similarity measure can compete with $C2$. This observation proves that selecting relevant click records from a user's QCW that is utilized in "Strategy 2" and "Strategy 4" can largely reduce the noisy information. Therefore, the similarity measure with the length factor alone can still gain considerably improvement. The results of our synthetic data set are consistent with those of real data set.

4.5.4.4 Effect of the Half Fading Parameter hf

We now evaluate the effect of the fading memory parameter hf on retrieval and re-ranking quality of the STAR framework. We plot the change of retrieval performance in Figure 4.7 given different values of the half fading parameter hf in Equation 4.2.

In other words, we indirectly discuss effects of different QCW sizes because we assign different weights for them, rather than truncating the whole QCW into different time cutoffs. Here click size is 25 and hierarchical semantic measure is "C2", which lets us focus on the issue that how the half fading factor hf influences retrieval accuracy. In Figure 4.7, for repeated IQs, the retrieval quality first becomes better (the precision scores increase) and then keeps relatively stable with addition of weights for long-term memories (increasing the value of hf). For semi-new IQs, the retrieval quality goes worse

TABLE 4.4: Effect of QCW size on efficiency

QCW size	20	40	60	80	100	120
Run time(ms)	1143.8	1151.4	1154.8	1163.7	1165.2	1167.3

(the precision scores show a gentle decreasing trend). This observation further confirms us that for semi-new IQs, the user's current search need can mostly inferred from the short-term memories. In this case, the user will do several searches. The successive searches implicitly include the user's current information need. Therefore, the short term memories play more important role on retrieval precision than the long term ones. On the other hand, repeated IQs work as long term user's interests and the user tries to get the previously retrieved information by issuing a same query.

We already discussed effects of different QCW sizes on retrieval effectiveness by changing the values of hf and we found that the QCW size has more effect on semi-new IQs than repeated IQs. The retrieval quality of repeated IQs keeps relatively stable while the retrieval effectiveness of semi-new IQs goes worse when increasing the value of hf. Therefore, Table 4.4 lists the effect of QCW size on retrieval efficiency of semi-new IQs by directly truncating the whole QCW into different number of records. The QCW size of each semi-new IQ is 116 since we extracted the QCWs of testing queries from the whole data set and we use "Strategy 4", click size=25 and hf=0.5. From Table 4.4, the larger number of records the QCW user profile contains, the more time each semi-new IQ runs. When QCW has all 116 records, the run time of each semi-new IQ is 1167.3ms. In each running, we will clean QCW first to let our algorithm run from the beginning of building each record in QCW to the ending of getting the re-ranking list. In real online application, we only need incrementally update QCW to store the new input query.

All the results show that re-ranking of search results through semantic based personalization actually can enhance the general search. We also confirm that there are two critical factors in our framework: (1) the query-to-query similarity which captures the long term search interests of a user (query dependent), and (2) the most recent search interest which reflects the short term search behavior of a user (time dependent). The two factors indicate that both short-term and long-term memories contribute to the improvement.

4.6 Summary

Chapter 3 and Chapter 4 investigated the feasibility and the impact such metadata collection may have on improving personalized search. Chapter 4 presented a query-centric STAR framework for selective utilization of user search behaviors and re-ranking and has

made two important contributions. First, we designed a novel user search profile called query context window (QCW) to record the search behavior of a user. Second, we developed a query-to-query similarity model and the fading memory based weight function. We showed how our STAR framework can carefully choose and select from the relevant click records the most useful user context for a given input query, and how we applied hierarchical semantic similarity measures in our re-rank strategies. We did extensive experiments to evaluate the influences of selective utilization of user profiles, the size of QCW, and the hierarchical semantic measures on retrieval accuracy. Our experimental results show that using ODP metadata to personalized re-ranking can effectively learn user-specific query-dependent personalization preference and significantly improve the accuracy of personalized search over the most existing personalized re-ranking schemes, e.g., the query and time independent re-ranking strategy in Chapter 3.

Chapter 5

Suggesting Related Keyword based Queries Using Web Access Logs

Chapter 3 and Chapter 4 have discussed how to build user profiles that store user preferences and how to use the use profiles to re-ranking the search results of search queries. A main problem, however, occurs for Web users is that if a user is not familiar with some domain, she might fail to choose terms at the appropriate level of representation for her information need, thus having difficulty in organizing and formulating her search query. In this case, suggesting related keywords is considered as an effective assistant to help Web users get their desired Web pages. The challenge is that search queries are especially short texts with only a couple of words which are not rich enough to achieve high suggestion accuracy. Specifically speaking, the conventional approaches of finding related queries rely on the common terms shared by two queries to measure the relatedness between them. Because of the very small term overlap between short Web queries, using term as a feature space cannot accurately estimate their relatedness. From Chapter 5 to Chapter 7, we study various alternative feature spaces to enrich a keyword based query, thus better representing the meaning of the query. We introduce how to get feature spaces from Web access logs (implicit relevance) in Chapter 5, from the top search results of queries (pseudo relevance) in Chapter 6, and from a trained probabilistic topic model in Chapter 6.

The rest of this chapter is organized as follows. Firstly, we introduce the background of keyword based query suggestion in Section 5.1. Then we describe how to get feature spaces from Japanese Web access logs and the relatedness definition between two

enriched queries in Section 5.2. After that, we address the details of experiment methodology in Section 5.3. Experiment results are discussed in Section 5.4. Lastly, we conclude this chapter in Section 5.5.

5.1 Introduction

Currently, some search engines give suggestions on the queries input by users, thus assisting them in rephrasing their queries to improve search quality. *Related search terms* in Google is based on the assumption that sometimes the best search terms for what a user is looking for are related to the ones the user actually entered. In the search box of Yahoo!, *Search Assist* compares an input query to searches all other Yahoo! users have composed and offers suggestions in real time. These services supplied by Google and Yahoo! highlight the importance of finding related queries. It is intuitive to rely on the common terms two queries share to measure the relatedness between them. A study of the log of a popular search engine [39] reported that most queries are about two terms per query. Since Web users typically submit very short queries to search engines, the very small term overlap between queries cannot accurately estimate their relatedness. For example, *New York Times* and *New York subway* have two terms in common, but the search results retrieved by the two queries are quite different. Moreover, it is possible that queries can be phrased differently with different terms but for the similar information needs. For example, *New York Stock Exchange* has no common terms with *Manhattan*, but Manhattan is the largest central business district in the United States and the site of New York stock Exchange. Given this problem, the technique to find semantically related queries (though probably dissimilar in their terms) is becoming an increasingly important research topic that attracts considerable attention.

Query enrichment is an effective method to find semantically related queries, which enriches the representation of a query by alternative feature spaces, instead of using terms in the query itself. Consequently, how to get suitable feature spaces becomes a key in this method. In this paper, given a search query, first we extract the Web pages accessed by users from Japanese Web access logs which store the users individual and collective behavior. From these accessed Web pages we usually can get two kinds of feature spaces, i.e, content-sensitive (e.g., nouns) and content-ignorant (e.g., URLs) which can be used to enrich search queries, thus better representing their meanings. Then, the relatedness between search queries can be estimated on their enriched representations, e.g., the overlap of their feature spaces. Our experiment results show that the URL feature space produces lower precision scores than the noun feature space which, however, is not applicable, at least in principle, in settings including: non-text pages like multimedia

(image) files, Usenet archives, sites with registration requirement, and dynamic pages returned in response to a submitted query and so forth. It is crucial to improve the quality of the URL (content-ignorant) feature space since it is generally available in all types of Web pages. The problem of the URL feature space is that even though two queries share no common URLs in their accessed Web pages, they may be related since Web pages with different URLs may semantically related.

We are inspired to find a novel content-ignorant feature space for query enrichment. The whole Web can be considered as a graph where nodes are Web pages and edges are hyperlinks. Recent research on link analysis has shown the existence of numerous Web communities [1] on the Web [30, 32, 46]. In this context, a Web community is a collection of Web pages with different URLs, but sharing common interest on a specific topic. A query can be enriched by the Web communities that the respective accessed Web pages belong to, instead of using the URLs of Web pages directly. Our Web communities are created from a Japanese Web page archive by only exploiting link analysis, thus they are regarded an alternative content-ignorant feature space. The proposed Web community feature space is novel, different from the traditional URL and noun feature spaces which are widely used in the literature [2, 5, 15, 19, 97, 99]. Experiment results show that the novel Web community feature space generates much better results than the traditional URL feature space. We also empirically analyze and compare the performance of the three feature spaces (i.e., URL, Web community, and noun) in a query recommendation system which suggests a list of related search queries given an initial input query. Users can utilize the suggested related search queries to tune or redirect the search process. We also reveal that using different feature spaces for query enrichment show different characteristics in finding related search queries.

5.2 Relatedness Definition based on Query Enrichment

Our goal is to find the related search queries given a current query input by a search engine user. Consequently, we need to measure the relatedness between queries and then recommend the top ranked queries. We will first discuss that what kind of feature spaces can be used for query enrichment and how to get them, and then we will give the definition of relatedness based on these feature spaces.

[1] In the field of social network, Web community is also used to mean a set of users having similar interests, which slightly differs from the definition in this paper.

5.2.1 Discussion

IIt is not solid to estimate the relatedness between two queries based on their term overlap. Clearly, the query-result-vectors present a better similarity metric than query term-vectors [75]. Our work follows the recent direction of using log data [2, 5, 19, 97]. It means that we augment a query by the feature spaces extracted from the Web pages accessed by search engine users. This log data based method utilizes the search history of users and is two-fold. On the one hand, the access information as implicit relevance can be automatically obtained. While collecting explicit relevance is labor-intensive since users have to manually give their judgments. On the other hand, one important assumption behind this log data based method is that all the accessed Web pages are *relevant* to the query, which is not as accurate as explicit relevance judgment in the traditional relevance feedback. It is true that choices made by a small number of users are likely to be unreliable. the large amount of information available in our Web access logs makes this less of a problem; we assume that users are more consistent in their choices of relevant Web pages than irrelevant ones. It is therefore reasonable to regard the accessed Web pages as relevant examples from a statistical viewpoint.

Given a query, we extract feature spaces from the accessed Web pages of our Japanese Web access logs. Each query is represented by its respective feature space. For example, if the feature space is the URL of a Web page, a query is enriched or represented by the URLs of all the accessed Web pages; if the feature space is the terms in a Web pages, a query is represented by the terms of all the accessed Web pages. Then, the relatedness between two queries can be estimated based on the similarity of their feature spaces, instead of their term overlap. The URLs of Web pages are content-ignorant and commonly available features, but the URL based query enrichment shows low precision in our experiments where the number of relevant queries is only a little more than the number of irrelevant queries. In addition, it is intuitive to use the contents of Web pages for query enrichment. However, content-sensitive based feature space (e.g., nouns) is not applicable in some cases as we discussed in Section 5.1. Therefore, we are motivated to improve the precision of finding related search queries using the URL (content-ignorant) based query enrichment. Next, we will introduce a novel content ignorant feature space, called Web community and how to create it using only linkage information between Web pages.

5.2.2 Creating Web Communities as Content-ignorant Feature Space

Directly using URLs as a feature space for query enrichment is simple but not very effective. It is understandable that even though the URLs of two Web pages are different,

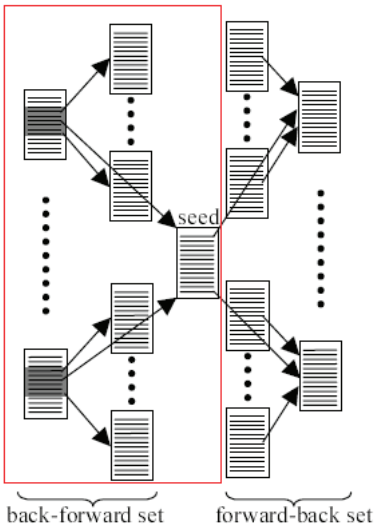

FIGURE 5.1: Vicinity graph for our *Companion−* algorithm

their contents may be related. We think if we can cluster related Web pages into a Web community which is a collection of Web pages sharing common interest on a specific topic, each query will be represented by the Web communities which the accessed Web pages belong to. Our experiment results show that the Web community based query enrichment largely improves the precision of finding related queries over the URL based one.

We create Web communities using only linkage information between pages without taking into account the contents of Web pages. This means that we need a Web page archive which stores the hyperlink information among Web pages. These Web pages are not only the accessed Web pages in our access logs, but also other Web pages on the Web during the period of collecting our logs. Thus, we can cluster Web pages using their linkage information. A large-scale snapshot of a Japanese Web page archive we used was built in February 2002. We crawled 4.5 million Web pages. A connectivity database is built to search outgoing and incoming links of a given Web page in the archive. To create Web communities, we used a manually maintained page list as a seed set which includes 4∼691 pages of companies, organizations and schools. And we extended the seed set by applying *Companion−* proposed by [95].

Here we briefly describe the *Companion−* algorithm that is an important step in our Web community extraction. First, we build a vicinity graph, which is a subgraph of the web around the seed. A vicinity graph is a directed graph, (V,E), where nodes in V represent Web pages, and edges in E represent links between these pages. The vicinity graph includes nodes that can be reached from the seed page by following incoming links then outgoing links (back-forward set), as illustrated in Figure 5.1. When following outgoing links from each node pointing to the seed in the back-forward set, not all the

links are followed but only R links immediately preceding the link pointing to the seed, and R links immediately succeeding the link. If a node has more than Nb incoming links, Nb links are randomly selected. In the right part of Figure 5.1, we also illustrate forward-back set which are built by following outgoing links then incoming links. The famous *Companion* proposed in [21] utilized both back-forward set and forward-back set. Experiment results show that our *Companion*− algorithm using only back-forward set can produce higher precision than *Companion* algorithm by 49.2%. We chose R to be 10, and Nb to be 2000. More result discussions are in [95].

After building vicinity graph, we assign weights to edges. To each edge, we assign two kinds of weights, an authority weight and a hub weight for decreasing the influence of a single Web page. The authority weight is used for calculating an authority score of each node, and the hub weight is used for calculating a hub score of each node. The notions of authority and hub are proposed by [44]. Simply speaking, an authority is a page with good contents on a topic, is pointed to by many good hub pages. A hub is a page with a list of hyperlinks to valuable pages on the topic, that is, points to many good authorities. We use the following weighting method which is also applied in *Companion*.

1. If two nodes of an edge have the same server part in their URLs, the edge has the value 0 for both weight.
2. If one node has n incoming edges from nodes in the same server, we assign each edge an authority weight of $1/n$.
3. If one node has m outgoing edges to nodes in the same server, we assign each edge a hub weight of $1/m$.

Then we calculate a hub score, $hub(n)$, and an authority score, $auth(n)$ for each node n in the vicinity graph, (V,E). The following is the process of the calculation, where $auth\_weight(n,m)$ and $hub\_weight(n,m)$ represent the authority weight and the hub weight of the edge from n to m, respectively. The computation steps are shown in Table 5.1.

Our *Companion*− algorithm computes authority scores for the Web pages in the seed set. Figure 5.2 illustrates a typical graph of authorities and hubs. In the right part of Figure 5.2, there are famous computer manufactures which are regarded as authorities. These authorities are densely linked by the hubs in the left part of Figure 5.2. For each seed, we select the top N authorities, and aggregate them into an extended seed set. We again apply the *Companion*− algorithm to each page in the extended seed set and build a new directed graph where an edge from a node s to an another node t exists when s derives t as one of the top N authorities. This directed graph is called the authority

TABLE 5.1: Calculation of Hub and Authority Scores

1:	Initialize $hub(n)$ and $auth(n)$ of each node n to 1.
2:	Repeat the following calculation until $hub(n)$ and $auth(n)$ have converged for each node n.
	For all node n in V ,
	$hub(n) \leftarrow \sum_{(n,m)E} \in auth(m) * hub\_weight(n,m)$
	For all node n in V ,
	$auth(n) \leftarrow \sum_{(n,m)E} \in hub(m) * auth\_weight(n,m)$
	Normalize $hub(n)$, so that the sum of squares to be 1.
	Normalize $auth(n)$, so that the sum of squares to be 1.
3:	Choose nodes with the N highest authority scores as results.

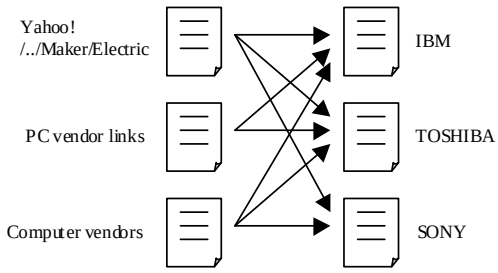

FIGURE 5.2: A typical graph of authorities and hubs

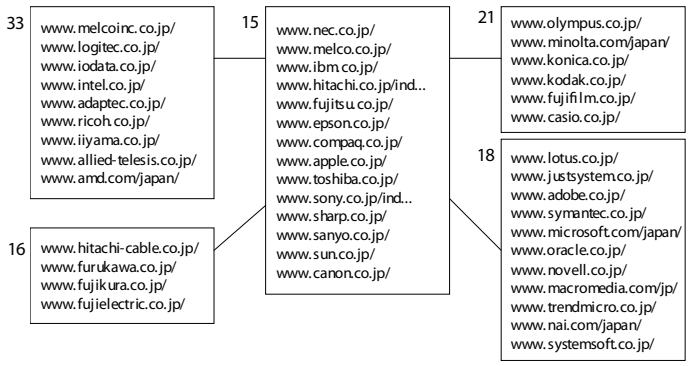

FIGURE 5.3: A part of our Web community chart

derivation graph (ADG). ADG built from our seed set includes 13,166 nodes and 70,201 edges.

The next step is to extract a symmetric derivation graph (SDG) from ADG. In this step, we put focus on the symmetric derivation relationships between nodes in ADG, that is, two nodes derive each other using *Companion−*. Using these derivation relationships between pages, we classify pages into communities based on complete bipartite graphs. Finally we automatically create 17 hundred thousand Web communities from one million selected pages. More technical details of creating Web community are in [95].

A part of our community chart is illustrated in Figure 5.3 where the number represents the community ID. In the center of Figure 5.3 there is a major community of computer

manufactures (15) surrounded by software (18), peripheral device (33 and 16), and digital camera communities (21). It is understandable that even if there are no common URLs of Web pages shared by two enriched queries, they may be related to some extent provided that some Web pages are clustered into a same Web community, e.g., the different Web pages in the computer manufactures (15) community. Each Web page in our data set is labelled by a community ID. Given the accessed Web pages of a query, their corresponding community IDs compose the Web community feature space of the query.

5.2.3 Content based Query Enrichment

We also study a content-sensitive feature space for query enrichment which can be a supplementary information to further enhance performance of finding related queries besides content-ignorant feature spaces (e.g., URL and Web community). As we know, nouns in a document can more accurately represent the topic described by the document than others. Therefore, a query is enriched by the nouns extracted from the contents of its accessed Web page sets. Our noun feature space is created using ChaSen, a Japanese morphological analyzer [2]. We have stored a Japanese Web page archive and completed the morphological analysis of all the Web pages in advance.

5.2.4 Definition of Relatedness

We discuss how to calculate relatedness between two queries enriched by a feature space. The goal of this chapter is to compare two content-ignorant feature spaces, i.e., URL and Web community, and also empirically study the performance of the noun feature space. Optimally, we had better design different similarity measures for different feature spaces in order to achieve the best performance for each feature space. Since we want to give empirical evidence as to how different feature spaces affect the quality of finding related queries, we need a similarity measure which can be easily applied to all the three feature spaces so that the experiment results based on this measure are reliable and fair.

Let $Q = \{q_1, q_2, \ldots, q_i, \ldots, q_n\}$ be a universal set of all search queries where n is the number of total search queries. Given the accessed Web pages of the query q_i, we extract three feature spaces: 1) URL: $U_{q_i} = \{url_1, url_2, \ldots, url_k\}$, 2) Web community: $C_{q_i} = \{com_1, com_2, \ldots, com_l\}$, and 3) Noun term: $T_{q_i} = \{t_1, t_2, \ldots, t_m\}$. The query q_i is enriched by one of three feature spaces, and then the relatedness between two queries can be estimated on the element intersection of their feature spaces. The popular Jaccard

[2] http://chasen-legacy.sourceforge.jp/

measure is intuitively suitable in this case, defined as

$$Jaccard(q_i, q_j) = \frac{|F_{q_i} \cap F_{q_j}|}{|F_{q_i} \cup F_{q_j}|}, \tag{5.1}$$

Here F_{q_i} denotes one of the three feature spaces, U_{q_i}, C_{q_i}, and T_{q_i}. Jaccard measure considers the number of common elements shared by two enriched queries.

From our log data, we can get the submit times of each query by various users, and each element in a feature space has its own occurrences, e.g., noun term frequency. Therefore, we are motivated to enhance the Jaccard measure by taking into account the frequency information. For each element in the feature space of a query, we assign it a weight which is the value of its occurrences in this feature space multiplied by the submit times of the query by various users. Then, for normalization, the calculated weight is divided by the sum of the weights of all the elements in the feature space of the query. Finally, we use the sum of the normalized weights of the common elements shared by two queries to estimate their relatedness. The normalized weight of each element in the feature space of a query considers the element frequency (occurrences) because more frequent elements are more likely to be better indicators for a query. Moreover, it also includes the popularity of the query (submit times by various users). Using this kind of normalized weight for relatedness computation means that we can find not only related queries based the quality of their feature spaces, but also popular queries favored by the majority of Web users.

We give a concrete example to understand our idea clearly. For the two search queries "bank" and "deposit", using the Web community feature space, we get their enriched representations as follow (number denotes Web community ID):
q_{bank}={37652,7968,1105,38251,9650,118781,140963,57665,150439,35392,37750,47811},
$q_{deposit}$={9650,140963,40786,38251,46820,37652,35392,1105}.
Based on our log data, the sum of the weights of all elements in the community space of "bank" is 113 and the sum of "deposit" is 13. Then, the normalized weights of these elements are
$C_{q_{bank}}=\{\frac{7}{113},\frac{3}{113},\frac{3}{113},\frac{2}{113},\frac{2}{113},\frac{1}{113},\frac{1}{113},\frac{1}{113},\frac{1}{113},\frac{1}{113},\frac{1}{113},\frac{1}{113}\}$,
$C_{q_{deposit}}=\{\frac{2}{13},\frac{2}{13},\frac{1}{13},\frac{1}{13},\frac{1}{13},\frac{1}{13},\frac{1}{13},\frac{1}{13}\}$.
For example, in $\frac{7}{113}$, 7 is the value of the occurrences of the Web community 37652 multiplied by the submit times of the query *bank* and 113 is the sum of weights of all the communities in $C_{q_{Bank}}$. The intersection of the two enriched queries includes six communities listed as follows:
$C_{q_{bank}} \cap C_{q_{deposit}}$={37652,1105,38251,9650,140963,35392}.
Therefore, the normalized weighted Jaccard score of the two queries, "bank" and "deposit" is: $\frac{0.142+0.615}{2} = 0.378$.

Notice that in the above example we only list the communities of $C_{q_{bank}}$ ($C_{q_{deposit}}$) which are not in a so-called excluded set. The reason is that the excluded set stores the *elements with high query frequency* in the three feature spaces. For example, the highly frequent elements of URLs are *Yahoo!, MSN, Google* and so on, and the highly frequent elements of nouns are *I, today, news* and so on. An element in a feature space is in the excluded set if the number of the test queries which feature spaces include the element are more than half of the number of all the test queries used in our evaluation. An element with high query frequency might be less informative to represent a distinct query.

Experiment results show that our relatedness definition can effectively find related queries. We would like to note that our work can easily incorporate other similarity and specificity measures. Due to the fact that this chapter focuses on an empirical study of different feature spaces for query enrichment to improve the quality of finding related queries, we omit the further discussion on more complex relatedness measures.

5.3 Experiment Methodology

We evaluate the effectiveness of our proposed Web community feature space, the URL and noun feature spaces and discuss the characteristics of the three feature spaces. In addition, because the techniques of query recommendation used in commercial search engines are usually confidential, it is difficult to do a quantitative evaluation. We design a query recommendation system which let us easily know the differences between our method and Google search engine by analyzing some concrete examples as case study.

5.3.1 Web Access Logs

Our Web access logs, also called "*panel logs*" are provided by *Video Research Interactive Inc.* which is one of Internet rating companies. The collecting method and statistics of this panel logs are described in [68]. Here we give a brief description. Panels (users) are randomly selected based on RDD(Random Digit Dialing), and are requested to install a software that automatically reports web access log to the server of Video Research Interactive. The panel logs consist of *user ID, access time of Web page, reference seconds of Web page, URL of accessed Web page* and so on. The data size is 10GB and the number of users is about 10 thousand.

In this study, we need to extract past search queries and related information from the whole panel logs. Figure 5.4 shows the details of a part of our panel logs. We notice that the URL from a search engine (e.g., Yahoo!) records the query submitted by a user,

UserID	AccessTime	RefSec	URL
1	2002/9/30 00:00:00	4	http://www.tkl.iis.u-tokyo.ac.jp/welcome_j.html
2	2002/9/30 00:00:00	6	http://www.jma.go.jp/JMA_HP/jma/index.html
3	2002/9/30 00:00:00	8	http://www.kantei.go.jp/
4	2002/9/30 00:00:00	15	http://www.google.co.jp/
1	2002/9/30 00:00:04	6	http://www.tkl.iis.u-tokyo.ac.jp/Kilab/Welcome.html
5	2002/9/30 00:00:04	3	http://www.yahoo.co.jp/
6	2002/9/30 00:00:05	54	http://weather.crc.co.jp/
2	2002/9/30 00:00:06	11	http://www.data.kishou.go.jp/maiji/
3	2002/9/30 00:00:08	34	http://www.kantei.go.jp/new/kousikiyotei.html
5	2002/9/30 00:00:07	10	http://search.yahoo.co.jp/bin/search?p=%C5%B7%B5%A4
5	2002/9/30 00:00:10	300	http://www.tkl.iis.u-tokyo.ac.jp/Kilab/Members/members-j.html

(a)

FIGURE 5.4: A part of our panel logs (Web access logs)

as shown in Figure 5.4(a). We extract the query from the URL, and then the access logs followed this URL in a session are corresponding Web pages browsed by the user. The maximum interval to determine the session boundary is 30 minutes, a well-known threshold [13], such that continuous accesses within 30 minutes interval are regarded as in a same session. Finally, we got about 125 thousand Japanese queries and 1 million accessed Japanese Web pages.

To obtain the Web community feature space, the other data set used is a Japaneses Web page archive crawled in February 2002. We already introduced it in the section where we discussed how to create the Web community feature space from the page archive. For each Web page in the access logs, we want to find its Web community ID in the archive. The time of Web page crawling for the Web page archive is during the time of the collection of access logs. Thus, there are some Web pages which are not covered by the crawling due to the change and deletion of accessed Web pages. We did a preliminary analysis that the URL overlap between the Web access logs and Web page archive is only 18.8%. After we chopped URLs to their hostnames, the overlap increases to 65%. This means that we find the Web community ID of an accessed Web page if the hostname of the URL of the accessed Web page belongs to a Web community. Moreover, our previous work [69] says that using the full URL path shows unsatisfactory performance and its average precision is lower than the noun feature space by 55.65%. We are now doing experiments to evaluate the effect of different path levels of a URL on experiment results. For example, we are using Web page's URL with one path element removed, two path elements removed and so on until we are left with just a hostname.

5.3.2 Evaluation Method and Kappa Statistics

Evaluation is not an easy job since there is an objective test data set for our problem at the current stage. Furthermore, we should evaluate the effectiveness of three feature spaces for each query, let alone parameter study. We invited nine volunteers(users) to do evaluation. They are our lab members who usually use search engines to meet their

TABLE 5.2: Test queries for evaluation

Test Query	#Accessed Pages	Group	Test Query	#Accessed Pages	Group
lottery	891	A	bank	113	C
ring tone	446	B	fishing	64	A
movie	226	C	scholarship	56	B
hot spring	211	A	university	50	C
soccer	202	B			

TABLE 5.3: Kappa and strength of agreement

Kappa	Strength of agreement	Kappa	Strength of agreement
0.00	Poor	0.41-0.60	Moderate
0.01-0.20	Slight	0.61-0.80	Substantial
0.21-0.40	Fair	0.81-1.00	Almost perfect

information needs. Nine test queries used in our evaluation are listed in Table 5.2. For each test query, we compute a recommendation list consisting of the top related queries according to their relatedness values on each of the three feature spaces. In addition, the relevance judgment has five levels, i.e., irrelevant, lowly relevant, relevant, highly relevant, and un-judged. Users chose one from the five relevance levels for each pair of a query and a recommended query.

After we get users' judgment results, we study the variability of user's categorical ratings to measure user disagreement which tell us how users classify individual subjects (queries) into the same category (relevance level) on the measurement scale. Kappa statistics is one of the most common approaches [86]. Kappa can be thought of as the chance-corrected proportional agreement, and possible values range from +1 (perfect agreement) via 0 (no agreement above that expected by chance) to -1 (complete disagreement). Table 5.3 provides a rough guide of what is a good agreement. We require users to answer questionnaires that supply two recommended queries per test query[3]. Because two users are grouped as a pair to compute a Kappa value, the total number of test pairs is 36 ($C_2^9 = 36$).

We collected the relevance judgment results of the nine test queries and the average of all Kappa values is 0.388. This value is not in the range 0.41-0.6, a moderate agreement in general. As we examined the contents of the questionnaire in detail, we noticed that several users chose the "un-judged" level in the recommended queries while most other users gave them a same evaluation (e.g., highly relevant). It is the reason that results

[3] 9 search queries * 2 recommended queries = 18 evaluated queries

TABLE 5.4: Evaluation results of the recommended queries with four relevance levels

Relevance	URL	Community	Noun
irrelevant	0.341	0.244	0.037
lowly relevant	0.107	0.089	0.043
relevant + highly relevant	0.445	0.611	0.843
	(0.106+0.339)	(0.131+0.480)	(0.135+0.707)
un-judged	0.107	0.056	0.078

in the fall of the Kappa value. Therefore, we deleted recommended queries which are evaluated as "un-judged" by users in the calculation of Kappa statistics. Then, the average value of our Kappa statistics became 0.41. In addition, there is a difference between the choice frequencies of the "relevant" and "highly relevant" judgments. We calculated the average Kappa value again by treating the two judgments as equivalence. Finally, we got a value of 0.508. To sum up, in terms of the statements in Table 5.3, the agreement in our results is moderate at 95% confidence level.

In our query recommendation, each search query will be suggested top 20 related queries. The total number of queries evaluated by users are 540 [4]. To alleviate the workload on an individual user, we divided the nine users to three group (i.e., A, B, and C) as shown in Table 5.2 and asked each group to give their relevance judgments on three queries. We can evaluate the effectiveness of different feature spaces by evaluation measures like $Precision$, MAP, and $NDCG$ [62] which are widely used for ranking problems, especially in Web search. Because the length of a term based query is much shorter than that of a Web page, search engine users can read the top ten or twenty recommended queries much more quickly than they browse and click the top Web pages one by one. In this case, we are interested in the number of related queries in a recommendation list, i.e., $Precision$, defined as the percentage of the number of queries judged as one of the five relevance levels over all recommended queries[5] given a feature space.

5.4 Evaluation Results and Discussions

5.4.1 Comparisons of Different Feature Spaces in Query Enrichment

The precision scores averaged by all the test queries are shown in Table 5.4 where we combine the results of highly relevant and relevant judgments. For example, The precision score of the "relevant + highly relevant" in the column "noun space" is 0.843

[4] 9 search queries * 20 recommended queries * 3 feature spaces = 540 evaluated queries
[5] 9 search queries * 20 recommended queries = 180 evaluated queries

which means the percentage of the number of queries judged as "relevant" or "highly relevant" over all recommended queries using the noun based query enrichment.

In Table 5.4, using the URL space, the number of recommended queries judged as "irrelevant" is more than using the community and noun spaces, while the number of recommended queries judged as "relevant + highly relevant" is less than other two spaces. Using the proposed Web community space, the number of recommended queries judged as "relevant + highly relevant" is more than using the URL space and the improvement is about 37.3%. While using the Web community space, the number of recommended queries judged as "irrelevant" is less than using the URL space and the improvement is about 28.45%. This means that our Web community based query enrichment is effective and largely improves the recommendation precision over using URL space. If the contents of Web pages are accessible, when the noun space based strategy is applied, about 80% of all recommended queries are evaluated as "relevant + highly relevant", while only about 3.7% of all recommended queries are evaluated as "irrelevant", which shows better results than the other two feature spaces. This result is easily understandable. Getting more information from Web pages to enrich a query can help us achieve better performance. As we discussed in Section 5.1, since there are several kinds of Web pages which are difficult to extract their contents, the content-ignorant feature spaces like URL and Web community is still of the essence. Our experiment results show that the proposed Web community feature space to enrich a query, which largely improve the recommendation precision over the URL feature space, and the noun features space can be a very useful supplementary source to further achieve higher precision.

The evaluation results of each individual test query are presented in Table 5.5. The community based strategy can give comparative results with the noun based strategy for some queries, e.g., "university", "hot spring", and "bank" queries; only for "lottery", the precision score of the URL based strategy is 0.833 which is close to that of the noun based strategy (i.e., 0.866). These results further confirm that the Web community based query enrichment is more effective than the URL based enrich, while noun based enrich is the best selection, but only in the case that we can get the contents of Web pages.

5.4.2 Case Study using Our Query Suggestion System

To conveniently do case study and compare our results with "Google Suggestion", we designed a query suggestion system which finds related past queries given an input query. Its interface is illustrated in Figure 5.5.

TABLE 5.5: Evaluation results of individual test queries

Relevance	Query	URL	Com	Noun	Query	URL	Com	Noun
irrelevant		0.100	0.450	0.100		0.250	0.083	0.000
lowly relevant		0.050	0.050	0.033		0.133	0.050	0.017
relevant	lottery	0.000	0.033	0.033	ring tone	0.050	0.067	0.067
highly relevant		0.833	0.450	0.833		0.567	0.800	0.900
relevant + highly relevant		**0.833**	0.483	**0.866**		0.617	**0.867**	**0.967**
un-judged		0.017	0.017	0.000		0.000	0.000	0.017
irrelevant		0.000	0.050	0.017		0.100	0.150	0.000
lowly relevant		0.067	0.050	0.067		0.133	0.067	0.000
relevant	movie	0.067	0.200	0.183	hot spring	0.383	0.233	0.100
highly relevant		0.550	0.483	0.567		0.233	0.550	0.700
relevant + highly relevant		0.617	**0.683**	**0.750**		0.616	**0.783**	**0.800**
un-judged		0.317	0.217	0.167		0.150	0.000	0.200
irrelevant		0.417	0.000	0.050		0.467	0.150	0.050
lowly relevant		0.000	0.017	0.017		0.283	0.117	0.167
relevant	soccer	0.117	0.050	0.067	bank	0.050	0.233	0.233
highly relevant		0.267	0.733	0.817		0.117	0.467	0.483
relevant + highly relevant		0.384	**0.783**	**0.884**		0.167	**0.700**	**0.716**
un-judged		0.200	0.200	0.050		0.083	0.033	0.067
irrelevant		0.767	0.617	0.000		0.717	0.600	0.100
lowly relevant		0.050	0.200	0.017		0.167	0.117	0.033
relevant	fishing	0.100	0.167	0.150	scholarship	0.067	0.183	0.200
highly relevant		0.000	0.000	0.717		0.033	0.100	0.617
relevant + highly relevant		0.100	0.167	**0.867**		0.100	0.283	**0.817**
un-judged		0.083	0.017	0.117		0.017	0.000	0.050
irrelevant		0.250	0.100	0.017				
lowly relevant		0.083	0.133	0.033				
relevant	university	0.117	0.017	0.183				
highly relevant		0.450	0.733	0.733				
relevant + highly relevant		0.567	**0.840**	**0.916**				
un-judged		0.100	0.017	0.033				

A user can input a search query in Figure 5.5(1) while the related queries recommended are divided into three parts according to different feature spaces. Then, the user can choose one recommended query to add or replace the initial query and submit the reformulated query to a search engine from a drop list as shown in Figure 5.5(2). Finally, the search results retrieved by the selected search engine are listed in the right part. Furthermore, in Figure 5.5(3), there are two slide bars which can adjust the lower and upper bounds of relatedness scores. For each feature space, the maximal number of recommended queries is 20. If the user wants more hints, she can click a button shown in Figure 5.5(4) to get more recommended queries ordered by their relatedness scores with the initial query.

Figure 5.5 presents the recommendation of the query "bank" as an example. Since in this study we utilize Japanese Web data, the corresponding English translation is in the bottom of this figure. If some queries are only available in Japanese, we give a brief English explanation. For example, the query "Mizuho" is a famous Japanese bank.

A Web community (a collection of Web pages sharing a related topic)includes Web pages with different URLs. The number of common elements of two queries enriched by Web communities may be more than that of URL based strategy. Using the query "bank" in

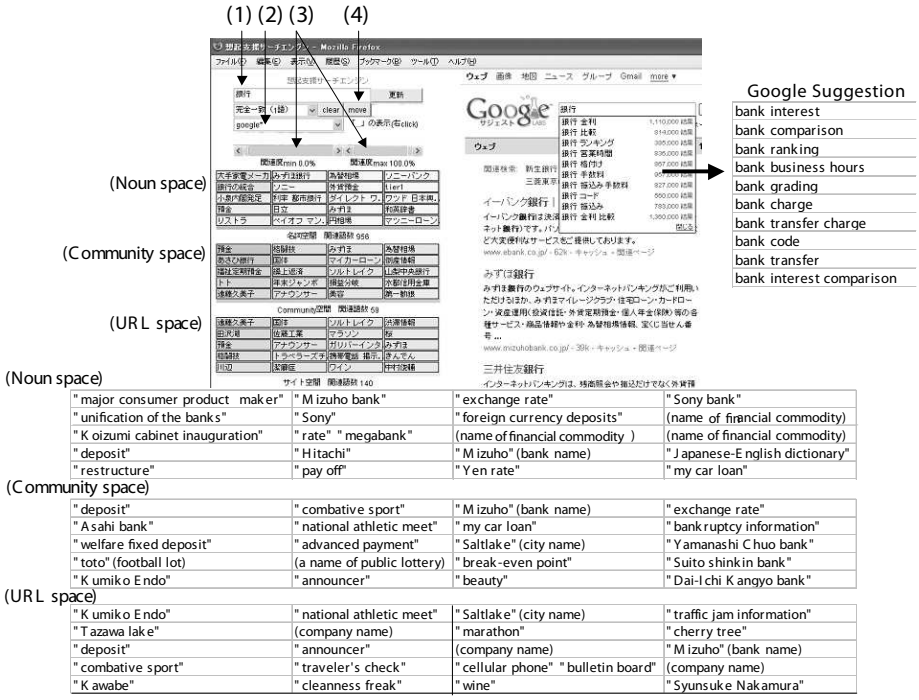

FIGURE 5.5: The user interface of our system

Figure 5.5 as an example, the community "city banks" has A bank and B bank. Using URL based strategy, the URLs of the homepages of the two banks are different, so they are not included in the common set, while using community based strategy, they belong to a same community, so they are included in the common set. For example, "Mizuho bank", "Suito shinkin bank", and "Yamanashi chuo bank" are popular banks in Tokyo. Moreover, our community extraction algorithm keeps the Web pages with high authority scores and deletes the Web pages with low authority scores in a community, which lays a foundation for finding related queries with high quality.

5.4.3 Differences with Google Suggestion

We illustrate the results of our case study in Figure 5.5, Figure 5.7, and Figure 5.6 to show the differences between our system and the query recommendation service provided by *Google Suggestion*. It is difficult to do a quantitative comparison with Google suggestion which has been trained on queries submitted to Google, while our tool has been trained on a different data set. It is not suitable to conclude which method is better, because two main factors play in this comparison: the different training data sets and the methods used which are likely different too. Therefore, we use some concrete examples as case study to show the differences.

In Figure 5.5 our system presents many good related recommended queries concerned with *bank* in the noun and community spaces such as "deposit", "my car loan", "toto"

Search Query "Fishing"

Noun space (F) means "fish name"			
"Shibayama" (F)	(F)	"lure" "Kiya-kyusyu city" "seabass"	"surff ishing" "Mihama word"
"(F) fishing" "Chiba city"	"rig" (F) "fishing"	"fishing" "lake Toyota" "point"	(F) (place name) "Hiroshima"
"January" "seabass" "minnow"	"Hyogo Meichoukai"	"Marukyu" "worm"	"fish guide"
"Zushi city" "point" (F)	"PEN reel" "Direct Marketing"	"rig" "fishing" (F)	"suma marine fishing park"
"Haneda Turimasa"	"fishing" "Yukuhashi city"	"rig" (F)	"Osaka bay" "point"
Community space			
"Kagoshima city"	"sea"	"part-time job"	"music"
"prayer"	"car"	"swimming beach"	"gsx1400"
"job offer"	"mah-jongg"	"job offer information"	"wine"
"christmas card"	"yahoo"	"job"	"house-moving"
"cad"	"chat"	"rental server"	"horse racing"
URL space			
"vector"	"Yasuo Uchiba"	"make links"	"BBS"
"computation" "capacity" "windows95"	"white day"	"Youichi Sasakura"	"cad"
"bulletin board" "license" "CAD qualification"	"books"	"flight"	"never7" (game title)
"printer"	"piano" "midi"	"golf"	"play station"
"simcity" (game title)	"smoked"	"constellation fortune-telling"	"mac"

The Results of Our System

Google Suggestion

fishing aa
fishing rig
fishing video
fishing game
fishing beginner
fishing blog
fishing weather
fishing how to knot
fishing winter
fishing bar

FIGURE 5.6: The results of our system and Google Suggestion for query: "Fishing"

Search Query "Soccer"

Noun space			
(football team)	"world cup" "image"	"wcup"	"England"
"fifa"	"Batistuta"	"soccer shop"	"Japan national team"
"world cup"	"World cup soccer"	"official shop"	"Inamoto player"
"Ilhan" (football player)	"Japan emperor's cup" "news flash"	"bikini"	"gossip" "Takayuki"
"Nakata Hide"	"soccer world cup"	"Beckham" "photo"	"cup fifa world"
Community space			
"Nabisco cup"	"soccer information"	"Tatsuya Enoki"	"Kazushi Kimura"
"soccer site"	"Masahiro Endo"	"England"	"world cup"
"Danish soccer"	"DF" "soccer"	"Europe" "soccer"	"jawoc"
"Alan Shearer"	"Passarella"	"Japan national team"	"senior soccer"
"2882017.0"	"toto" (football lot)	"f" "marinos" (football team)	"asian cup"
URL space			
"Japan emperor's cup"	"Koizumi cabinet"	"Byrom Inc"	"princess aiko"
"sports news"	"world cup"	"Sarah Hughes"	"Yuko Yamaguchi"
"yahoo! nba"	"woman pole vault" "dragila"	"professional baseball flash"	"soccer world cup"
"toto" (football lot)	"Japan national team"	"Ehime Maru"	"Nekohachi Edoya"
"f1"	"Ikko Tanaka"	"senior soccer"	"christen" "princess aiko"

The Results of Our System

Google Suggestion

soccer transfer
soccer Japan national team
soccer uniform
soccer transfer information
soccer video
soccer spike
soccer news
soccer rule
soccer senior high school
soccer blog

FIGURE 5.7: The results of our system and Google Suggestion for query: "Soccer"

and so on that are not given by *Google Suggestion*. There are better recommended queries using noun space than *Google Suggestion* in the example of *Fishing* as shown in Figure 5.6. For example, "fishing aa" recommended by "Google Suggestion" is not highly relevant to "fishing" while the noun based strategy gives "Marukyu", "worm", "fish guide" and so on which are more related to "fishing". We observe that those suggested queries returned from Google are rather similar in their terms and usually share common term. For instance, if a user searches for *Soccer*[6] in Google Japan, the following related queries are presented: *soccer transfer*, *soccer Japan national team*,

[6]In general, Japanese say *soccer* not *football*

soccer uniform, *soccer video*, and so on. The recommended queries by our system give some football player such as *Beckham* and *Batistuta*, as shown in Figure 5.7.

5.5 Summary

In spite of the improvement of searching accuracy with the development of technologies, it is not always true that search engine users can hit upon the proper search queries. In this paper, we studied the problem of how to find related search queries which can help users refine their original queries and get their desired Web pages. To enrich the representation of search queries, directly using the URL feature space is simple but not very effective. It is understandable that even though the URLs of two Web pages are different, their contents may be related. We proposed a novel feature space called Web community. Topic-related Web pages are clustered into a Web community and each query is represented by the Web communities which its accessed Web pages belong to. The relatedness between queries based on the common Web communities they share produces much higher precision than the traditional URL feature space. In addition, although the noun feature space can find more related search queries than the URL and Web community spaces, it is not generally available in the case of non-text pages. Therefore, the noun feature space can be used as a supplementary to further enhance the performance. We also provided empirical evidence as to how different feature spaces (i.e., noun, URL, and Web community) affect the results of finding related queries by using large-scale Web access logs and Japanese Web page archive. Moreover, a query recommendation system was devised to compare our results with "Google Suggestion" and conveniently analyzed the differences of the three feature spaces of query enrichment.

Chapter 6

Suggesting Related Keyword based Queries using Query-URL Bipartite

In Chapter 5 we have studies the implicit relevance query suggestion which, however, has its drawback. If the input query by a user is totally new to current Web logs, its click information is not available for similarity computation. In this case, we may think of using pseudo relevance, e.g., the top search results of queries. This chapter works on extracting feature spaces from the pseudo relevance. Like what we have done in Chapter 5 we are also interested in improving the suggestion precision of the basic content-ignorant feature space, i.e., URLs of top search results. We present a **Q**uery-**U**RL **Bi**partite based query re**C**ommendation approach, called QUBiC. It utilizes the connectivity of a query-URL bipartite graph to recommend related queries and can significantly improve the accuracy and effectiveness of the conventional pairwise similarity based approach.

The remainder of this chapter is organized as follows. We introduce our research motivation in Section 6.1. An overview of our approach is presented in Section 6.2. We describe the three phases of the QUBiC adaptive approach for query recommendation in Section 6.3, Section 6.4, and Section 6.5. An extensive experimental evaluation on the effectiveness of the QUBiC approach is reported in Section 6.6. We conclude the chapter in Section 6.7.

6.1 Introduction

As we discussed in Chapter 5, Web users, typically submit very short queries to search engines [39]. Therefore, the very small term overlap between queries cannot accurately estimate their relatedness. As a concrete example, we take two queries, "IRS (Internal Revenue Service) form" and "file taxes online". Although they have no terms in common, both of them concern the application of filing taxes. The relatedness between the two queries can be induced from the overlap of the two lists of search results returned. Thus, the query result-vectors are often a better similarity metric compared to query term-vectors [75]. Both URLs and contents of search results can be used as feature spaces to compute query-to-query similarity. Our previous experiences in Chapter 5 show that the URL based suggestion usually shows worse performance than the content based suggestion. In this chapter, we describe an adaptive approach to improve the precision of the URL based suggestion which is different from the approach proposed in Chapter 5.

To utilize the query result-vectors, one obvious approach is to represent queries and their result URLs as a bipartite graph with edges connecting queries on one side of the graph to the corresponding URLs of search results returned by a search engine on the other side of the graph. It is clear that related queries can be found by examining the collection of queries and URLs in the query-URL bipartite graph. Queries that are more strongly connected to each other are considered more similar (related). Different approaches can be used for finding a collection of similar queries. From a graph theoretical viewpoint, extracting the collection of most related queries and URLs can be approximated by the problem of partitioning a bipartite graph [5, 104] and then finding the maximum biclique, one of the well-known NP complete problems in the literature [20]. The problem of partitioning a bipartite graph can also be addressed by clustering techniques [2, 36, 97]. Both bicliques and clusters are groups of related queries. Queries in different groups are regarded as unrelated, or at least much less related than queries in a same group. After finding groups of similar queries, the next challenge is, given a query, how to devise a ranking mechanism to order the related queries in a group, and making query based recommendations.

Our QUBiC system fulfills two tasks: 1) finding groups of related queries, and 2) ranking queries in the same group as an input query by proceeding in three phases.

The first phase is the preparation of queries and their search results returned by a search engine for constructing a query-URL bipartite graph. A query is represented in terms of the corresponding set of URLs returned by the search engine in responding to the query, instead of the set of terms used in the query keyword list, thus realizing query result-vectors.

On the second phase, we generate query affinity graph based the similarity measure between pairs of queries. We use query-URL vector model to measure similarity between queries, instead of using query-term vector model. We argue that the query-URL vector better represents the meaning of a query, given the frequency and the amount of Web queries either phrased in the same list of keywords but meant for different results by different users, or phrased with different terms but meant for the same information need. Connected components can be extracted from the query affinity graph and each connected component is a group of related queries, thus completing the first task of the QUBiC system.

In the third phase, we discuss that it is insufficient and local that the naïve approach directly uses the similarity scores of pairs of queries computed in the second phase. Such naïve approach is widely adopted in the literature of mining related or similar items (e.g., queries, pages, etc.) [2, 11, 33, 67, 97]. We propose to utilize the monotonicity of the merging distances of a hierarchical agglomerative clustering (HAC).

By using HAC based algorithms we can globally capture the diffusive transition of similarity on each connected component to rank queries. Furthermore, instead of fixing the degree of similarity of queries based on their query-URL vector similarity, the QUBiC system adaptively controls the output of different lists of related queries in terms of the level of users' satisfaction.

6.2 QUBiC System Overview

The QUBiC system is designed to implement an adaptive approach to query recommendation based on query-URL bipartite graph. Figure 6.1 shows a sketch of the QUBiC system architecture. As illustrated in the rightmost three boxes in Figure 6.1), the proposed query recommendation approach consists of query-based recommendation preparation to build the query-URL bipartite graph, generating query affinity graph to discovery groups of similar queries, and the ranking of similar queries by taking into account both the propagation of the similarity and the subjectivity of the similarity.

Because the queries other users previously entered may be related to the information need of current user, finding related queries will give hints to an individual user. In the first phase, for each query in a set of past queries, we retrieve the top N search results from a search engine to represent this query (the parameter N will be studied in experiment part). The query-URL bipartite graph is built by creating edges between a query and its corresponding search results, as shown in Figure 6.2. Although it is intuitive to represent queries by their terms, we have argued that this approach fails

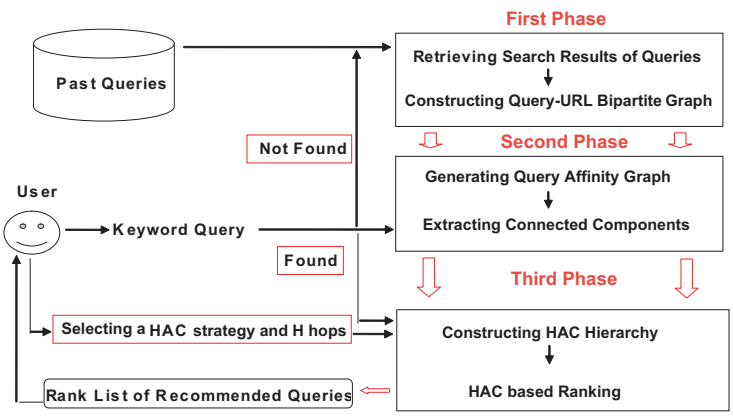

FIGURE 6.1: QUBIC architecture

to handle several typical cases in Web queries, for example, queries using the same keyword list but meant for different set of results by different users and queries with different terms but for the same information needs.

The second phase generates query affinity graph (QAG) and extracts connected components from QAG, thus forming groups of similar queries. Instead of using graph-theoretical approach of finding maximum biclique on the bipartite graph built in the first phase, we construct a query affinity graph (QAG) using a query-URL based distance (similarity) measure. A system defined parameter δ is introduced to control the level of query similarity to be considered, thus reducing the set of candidate queries in QAG for considering most relevant ones. Last, connected components are extracted from QAG. Each connected component is regarded as a group of queries with high relevance or similarity in terms of their query-URL vector similarity measure.

In the third phase, given an input query, we need to rank queries in the same group as the input query, to produce the best adaptive query recommendation. The naïve approach to ranking similar queries is to use the query-URL vector based similarity scores, which fails to properly capture the propagation of similarity in the query affinity graph and the subjectivity of similarity from users' requirements.

For example, in Figure 6.3, queries are connected by weighted edges. The weights are directly computed by some query-URL vector based similarity measure (see Section 6.4 for detail). The related queries of the query q_2 are q_1 and q_4, which are adjacent nodes to q_2 in the query affinity graph. The similarity score between un-adjacent nodes (e.g., q_2 and q_3) is zero, which, however, does not mean the two queries are definitely unrelated. We think that the ranking of similar queries for query recommendation should take into account the similarity propagation in the sense that similarity scores may propagate from query to query, exhibiting implicit relatedness between un-adjacent nodes in QAG (e.g., q_2 is related to q_3 via q_1 in our running example).

The monotonic merging operations of hierarchical agglomerative clustering (HAC) strategies generate the dendrogram (also called cluster tree) for displaying the cluster membership between data points. Based on this kind of tree structure, we argue that a tree distance based ranking can better address the problem of similarity propagation. Moreover, the flexibility of a HAC strategy allows the lists of related queries to be changed from user to user and query to query, thus adaptively recommending related queries on demand.

The general work flow of our system with the three phases consists of offline and online processings. Because we have a collection of queries, the offline processing can be completed as system initialization. We only need to apply the first and second phases to these queries and obtain groups of related queries. In the online processing, a user inputs a keyword query to our system. If the input query can be found in one of these groups of related queries, the third phase ranks each query in this group and generates a recommendation list. If the input query cannot be found, the whole three phases are executed one by one. Thus, additional cost is incurred. We need to add this new query and its search results to the original query-URL bipartite graph, re-compute the similarity (distance) scores between the input query with all the queries in the query collection, and update the query affinity graph. We will discuss some optimization techniques to reduce the updating cost, especially when updating the query affinity graph in Section 6.4.

6.3 Query-based Recommendation Preparation

Web search engines are useful to find alternative information sources for more accurate representation of a query than query-term vectors. Given a query, a Web search engine will return relevant Web pages (search results) according to its own rank mechanism. Many advanced Web searching techniques have been developed and used in commercial Web search engines, e.g., Google and Yahoo!, which can retrieve search results with high relevance to an input query, but also high page quality like Pagerank [70] and HITS [44]. Therefore, we are interested in the top search results in a retrieved list after submitting a query to a search engine. From these returned search results, usually we can extract their contents and their URLs to enrich the query. The content of a Web page, however, is not applicable, at least in principle, in settings including: non-text pages like multimedia (image) files, Usenet archives, sites with registration requirement, dynamic pages returned in response to a submitted query and so forth. In this chapter we study the utilization of the URLs of returned search results to represent a query since URLs are generally available in all types of Web pages.

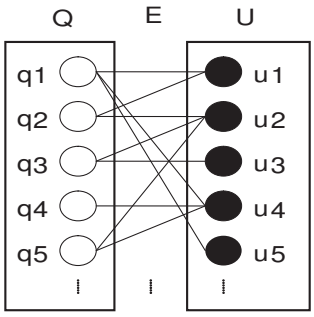

FIGURE 6.2: Query-URL bipartite graph

TABLE 6.1: Query-based recommendation preparation phase

Input: a set of queries .		
Output: query-URL bipartite graph.		
1.	Submit each query to a search engine (e.g., Google) and get the top N search results returned;	O($\|Q\| \cdot t$)
2.	Extract the URLs of all the search results	O($\|U\|$)
3.	If a query q appears with an URL u in the set of its search results, place an edge in the bipartite graph (BG) between the corresponding vertices in Q and U;	O($\|E_{BG}\|$)

Now a query is associated with the URLs of the retrieved search results. In the query-based recommendation preparation of our QUBIC system, the query-URL relationship can be intuitively represented as a bipartite graph. A bipartite graph, also called a bigraph, is a special graph from which the set of vertices can be decomposed into two disjoint sets such that no two vertices within the same set are adjacent. In the mathematical definition, a simple undirected graph G: $=(Q \cup U,\ E_{BG})$ is called bipartite if Q and U are disjoint sets, where Q and U are the vertex set and E is the edge set of the graph. This graph is used as our original model where Q is a set of queries, the U is a set of URLs, as shown in Figure 6.2. An edge e connects a query q and an URL u, if the URL u is returned by a search engine on the query q.

In the context of QUBIC, we propose to recommend queries based on the inter-relationship of their corresponding URLs. As a by-product, related Web pages could be mined as well by applying the proposed algorithm on the side of the URLs with a relatively small modification.

The pseudo code of the first phase is listed in Table 6.1. The run time cost of this phase is mainly dependent on the first step which time complexity is O($|Q| \cdot t$) and t is the average time cost of the response speed of a search engine and the network delay. The whole first phase can be done in offline as a preparation for the next two phases. Even if an input query is not in the current query set, we only need add the query as a node

TABLE 6.2: Generating query affinity graph phase

Input: query-URL bipartite graph. **Output:** connected components.	
4. Compute the distance scores between each pair of queries according to one of two similarity measures;	$O(\|Q\|^2)$
5. Link every pair of queries with a weighted edge if their distance score is smaller than δ, thus producing the query affinity graph (QAG).	$O(\|E_{QAG}\|)$
6. On the basic initialization of the disjoint-sets structure [18], each node in QAG is in its own set;	
7. The connected components are calculated based on the edges in QAG, so update the disjoint-sets structure when each edge is added into the graph;	
8. Extract the connected components;	$O(\|Q\| + \|E_{QAG}\|)$

to Q and the URLs of its search results to U. The total number of added edges is the number of returned search results. Therefore, the time cost of updating the bipartite graph is acceptable in the first phase.

6.4 Generating Query Affinity Graph

In this section we describe the design of the generating query affinity phase of QUBIC, which applies a URL-based similarity(distance) measure for all pairs of queries. The pseudo code of this phase algorithm is listed in Table 6.2.

6.4.1 URL-Vector Based Similarity Measures

As discussed in the first phase, the query q in Q is represented by the list of URLs returned as search results. Furthermore, let $w_{q,u}$ be the weighted value associated to the query-URL pair (q, u). $w_{q,u}$ is 0 if u is not included in the list of URLs returned by a search engine in response to the query q, otherwise $w_{q,u}$ is 1 in an unweighted strategy. $w_{q,u}$ can also be in the range [0,1], determined by different weighting strategies in different similarity (distance) measures. Therefore, the query q is denoted by $w_{q,u_1}, w_{q,u_2}, \cdots, w_{q,u_{|U|}}$ where $|U|$ is the number of URLs in U. Using the notations described above, we define the different similarity measures between queries. Although they are called similarity measures, they, in fact, can be used as distance scores by a simple transformation. We use *similarity* and *distance* interchangeably to conveniently describe our problem in different contexts. In this chapter we discuss three distance

functions that utilize the URL-bipartite graph to compute the distances between two queries, quite different from the term-based traditional measures.

All the three distance functions rely on the overlap of the search result URLs of two queries with unweighting or weighting function on each URL. We can utilize the different path levels of a URL to get the overlap. Generally, using the full URL of a search result will result in smaller overlap and higher precision than using its hostname. We experimentally calculated the overlap of hostnames of top 10 search results between queries and about 80% of the pairs of queries had no common hostnames (details in Section 6.6), not mention the full URL. We can increase the number of top search results to get larger overlap of full URLs, but it will degrade the precision of measuring the query-to-query distance (similarity) since the top search results are better dictators of a query than the bottom ones. Therefore, we use the hostname of a URL to represent a search result, instead of its full path. From now the denotation URL means *hostname* if no specific explanation.

(1)**Jaccard Similarity**

Jaccard similarity is an intuitive and popular measure, defined as

$$Jaccard(q_i, q_j) = \frac{|P(q_i) \cap P(q_j)|}{|P(q_i) \cup P(q_j)|}. \tag{6.1}$$

Here q_i and q_j are queries, $P(q_i)$ and $P(q_j)$ are the two sets of URLs returned by a search engine in response to the two queries q_i and $q_j$respectively. The value of the defined distance between two queries lies in the range [0, 1]: 1 if they are exactly the same URLs, and 0 if they have no URLs in common. For example, if $P(q_1)$={u_1, u_3, u_4} and $P(q_2)$={u_1, u_2}, the Jaccard similarity between q_1 and q_2 is 0.25. We use $d(q_i, q_j) = 1 - Jaccard(q_i, q_j)$ as Jaccard distance for the clustering computation in the third phase of our system. This definition is an unweighted strategy for URLs which only count the number of commmon URLs shared by two queries. To consider the frequency of u that occurs in a query and the number of queries containing u in their results, we propose the following two different weighting methods.

(2)L_1 **Norm**

Specifically, the weighting function for each URL is defined as:

$$w_{q,u} \equiv p(u|q) = \frac{n(u|q)}{\sum_{u \in U} n(u|q)}, \tag{6.2}$$

where $n(u|q)$ is the number of occurrences of the URL u in the query q. For example, if a URL (hostname) occurs five times in the search results of a query, we could say

that this website can be more informative to represent the query than other websites with lower frequency. Each query has its own URL (hostname) distribution because of different search results.

In document clustering, a natural measure of similarity of two documents is the similarity between their word conditional distributions. Roughly speaking, documents with similar conditional word distributions would belong to the same cluster. This idea was first introduced in [71] and was called "distributional clustering". Similarly, we would like to recommend related queries with similar conditional URL distributions. L_1 norm (or the variational distance) is common to measure the distance between distributions, defined as

$$L_1(p(u|q_i), p(u|q_j)) \equiv \frac{\sum_{u \in U} |p(u|q_i) - p(u|q_j)|}{\sum_{u \in U} p(u|q_i) + \sum_{u \in U} p(u|q_j)} \,. \tag{6.3}$$

We also have experimentally evaluated Ward and Jensen-Shannon (JS) divergence [58] measures which showed no better performance than L_1 norm on average. Therefore, we only discuss the L_1 norm in this chapter. However, popular sites like Google and Yahoo! may be returned within the search results of a query, which will be noisy irrespective of the content of the query. If two queries both retrieve such popular sites, some relatedness between them will be deduced, even though they may be completely unrelated. We think to scale down the coordinates of hostnames that occur in many result lists of queries, and put forward a URL-based cosine similarity in the following.

(3)**Cosine Similarity**

Similar to the traditional *tf*idf* weighing method, a URL-vector based $w_{q,u}$ is defined as:

$$w_{q,u} = (1 + ln(1 + ln(n(q,u)))) * ln(1 + |Q|)/m_u \,, \tag{6.4}$$

where $n(q,u)$ is the number of times the URL u occurs in the query q, $|Q|$ is the total number of queries in the query set, and m_u is the number of queries containing u. The similarity of each pair of query vectors is calculated using the following cosine formula:

$$Cosine(q_i, q_j) = \frac{\sum_{u=1}^{|U|} w_{q_i,u} * w_{q_j,u}}{\sqrt{\sum_{u=1}^{|U|} (w_{q_i,u})^2} \sqrt{\sum_{u=1}^{|U|} (w_{q_j,u})^2}} \,. \tag{6.5}$$

The distance between q_i and q_j is simply computed by $d(q_i, q_j) = 1 - Cosine(q_i, q_j)$. Intuitively, more frequent URLs (hostnames) are more likely to be better indicators for a query (Equation 6.2 in L_1 norm); while URLs (hostnames) with higher query frequency might be less informative to represent a distinct query, such as the homepages of popular search engines, and so on.

In L_1 norm and $Cosine$, u is weighted based on its occurrences in a query and its query

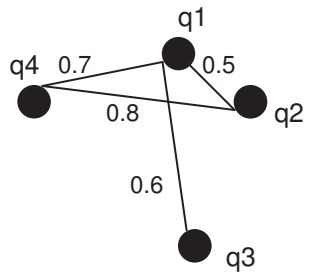

FIGURE 6.3: An example of query affinity graph

frequency. At first glance, for each q and u, $w_{q,u}$ can easily be valued by the rank of u for q. Using the rank values of URLs will take into account the ordering of search results when computing similarity scores. If the query q_i is of the same search results and the same ordering of search results as the query q_j and a user is not satisfied with the results of q_i, the results of q_j ought to be unsatisfactory. Thus, our weight methods do not consider the rank values of search results. We will discuss this problem further in Section 6.5.3.

6.4.2 Generating Query Affinity Graph

For each pair of queries, we add an edge in the query affinity graph (QAG) with the queries in Q as the nodes if their result URL sets have at least one common URL (see Figure 6.3 for an example). Then, we compute a distance (similarity) score for each pair of queries in QAG using one of the above three measures. The edge between two queries is weighted by the distance score (Step 5). In QUBIC system, we use the system supplied parameter δ (e.g., $\delta = 0.85$) to remove the edges between queries whose distance scores are larger than δ when extracting connected components (step 8). This helps to keep the overall data within a reasonable size. The extraction of connected components from an undirected graph is calculated in Step 6 $\sim$ Step 7 and readers can refer to [18] for detail. The time complexity for calculating the connected components is only slightly larger than $O(|Q| + |E_{QAG}|)$ where $|Q|$ is the number of queries and $|E_{QAG}|$ is number of edges in the QAG.

Based on the QAG, two queries are related if there are paths from one to another. Under this definition, we generalize the concept of relatedness in the sense that given the query q_2 in Figure 6.3, its related queries include adjacent nodes (e.g., q_1 and q_4) and un-adjacent nodes (e.g., q_2) as long as they are reachable from the given query following a path in QAG.

6.4.3 Optimization of Similarity Computation and Update of Query Affinity Graph

The first and second steps are used in the offline process for system initialization and in the online process when an input query is not in our query collection. Given that the query-URL bipartite graph is stored in our system, should we have to compute similarities for all pairs of queries that will count a quadratic number of values in the offline process? Moreover, additional cost is incurred in the online process when the input query is new to our system, since this new query and its search results have to update the original query-URL bipartite graph and the query affinity graph. As we discussed in Section 6.3, the cost of updating the query-URL bipartite graph is acceptable. Should we compute similarity scores for all the stored queries with a new query to update the query affinity graph?

We introduce an optimization technique to reduce the cost of updating the query affinity graph, especially similarity computation. Recall that we are interested in pairs of queries whose similarity is above a specified threshold, a high quality collection. The recent work [4] addressed this scalability issue without relying on approximation methods or extensive parameter tuning, and described different monotone minimum constraints for different similarity measures before computing the similarity scores. Here we use Jaccard measure as an example to explain their basic idea. For simplicity, let δ'=1-δ. We set up the query affinity graph using pairs of queries whose similarity is above δ'. The following inequalities are established:

$$\begin{aligned} Jaccard &= \frac{|P(q_i) \cap P(q_j)|}{|(P(q_i)| + |P(q_j)| - |P(q_i) \cap P(q_j)|} \\ &\leq \frac{|P(q_i)|}{|(P(q_i)| + |P(q_j)| - |P(q_i) \cap P(q_j)|} \\ &\leq \frac{|P(q_i)|}{|(P(q_i)| + |P(q_j)| - |(P(q_i)|} \\ &\leq \frac{|P(q_i)|}{|P(q_j)|} \leq \delta' . \end{aligned}$$

The above equation tells us if the $|P(q_i)|/|P(q_j)| \leq \delta'$, their similarity score does not need to be stored to reduce the amount of candidate pairs. Therefore, we can sort queries in the decreasing order of $|P(q)|$ to save on computation time. This preprocessing means that if $|P(q_i)|/|P(q_j)| \leq \delta'$ is met, the queries p_i to last query with the smallest number of search results can be skipped without doing similarity computation with p_j, thus accelerating our approach. Based on the similar idea, Bayardo et al.[4] described how to get constraints for *Cosine*, *Dice*, and other popular measures in details (L_1 norm can be regarded as a kind of *Dice* measure). Therefore, we can sort our data set according to

particular criteria such as vector size (e.g., $|P(q)|$) to improve the system computation performance.

6.5 Relevance based Query Ranking and Recommendation

The primary task of the third phase is to rank queries in the same group as an input query. We first discuss the design of our ranking methods and then describe how to adaptively output a recommendation list of related queries.

6.5.1 Our Ranking Methods

Our ranking methods are HAC-based, and their respective clustering results (i.e., dendrograms) can estimate the relatedness between queries. Our idea is based on the hierarchical structure of the dendrogram. Hierarchy is more informative than the unstructured set clusters in flat clustering algorithms like k-means and EM. As we know, HAC treats each query as a singleton group at the beginning, and then merges pairs of groups iteratively until all groups have been merged into a single group structured as a hierarchy that contains all queries.

Furthermore, HAC has an interesting property that distance measures associated with successive merge operations can be monotonic. If d_1, d_2, $\cdots$, d_k (the definition will be expressed soon) are successive combination distances of an HAC strategy, then $d_1 \leq d_2 \leq \cdots \leq d_k$ must hold. Urged by the monotonic property, the pair of queries that has the shortest distance will be merged first. At each remaining step, the next closest pair of queries (or groups) should be merged. The sequence of merge operations scores the relevance of two queries and then such relevance is encoded as a dendrogram (a cluster tree) where edges are weighted by combination distance. We make use of this kind of tree structure to produce an ordered list of related queries for a specific query and the tree distance based ranking can better address the problem of similarity propagation.

The monotonic property is desirable in our context. However, not all HAC algorithms are monotonic. Lance et al. [49, 50] proved that single-linkage and complete-linkage strategies are monotonic by definition. Moreover, they introduced a generalized recurrent formula including all special cases, defined as:

$$d_{hk} = \alpha_1 d_{hi} + \alpha_2 d_{hj} + \beta d_{ij} + \gamma |d_{hi} - d_{hj}| \,, \tag{6.6}$$

where the parameters α_1, α_2, β, and γ determine the nature of the strategy. (h), (i), and (j) are three groups, containing n_h, n_i, n_j elements respectively with inter-group

distances already defined as d_{hi}, d_{hj}, d_{ij}. They further assume that the smallest of all distances still to be considered d_{ij}, so that (i) and (j) fuse to form a new group (k), with n_k $(=n_i+n_j)$ elements. The strategy is necessarily monotonic so long as $\alpha_1+\alpha_2+\beta \geq 1$ and $\gamma=0$.

Previous work usually utilized HAC strategies to find clusters, while our approach designs a rank mechanism based on the clustering result (i.e., dendrogram) of HAC strategies. No much experience can be learned from previous work. Therefore, we adopt three popular monotonic HAC strategies for empirical study and propose another flexible HAC strategy for adaptive outputting recommendation lists. All the following HAC strategies can be derived from Equation 6.6.

(1)**Single-linkage strategy (SL):**
The single-linkage strategy merges the two clusters with the smallest minimum pairwise distance, defined as:

$$\begin{aligned} d_{hk} &= MIN[d(q_h, q_k)] \\ &= \tfrac{1}{2}d_{hi} + \tfrac{1}{2}d_{hj} + 0d_{ij} - \tfrac{1}{2}|d_{hi} - d_{hj}|\,, \end{aligned}$$

where although $\gamma <> 0$ does not satisfy the monotonicity constraint, the single-linkage strategy is monotonic by definition [49, 50]. However, in the single-linkage clustering, the similarity of two clusters is the similarity of their most similar members. This criterion is local [62]. We pay attention solely to the area where the two clusters come closest to each other. Other, more distant parts of the cluster and the clusters' overall structure are not taken into account. Therefore, it is easily affected by outliers and a chain of points can be extended for long distances without regard to the overall shape of the emerging cluster. This effect is called chaining. Since its merge criterion is strictly local, other strategies are studied for comparison. An alternative monotonic clustering is the complete-linkage strategy which is the opposite of the single linkage strategy, but unsuitable for our problem. For example, in Figure 6.3 there are $d(q_1, q_2) = 0.5$, $d(q_1, q_3) = 0.6$, and $d(q_2, q_3) = 1$. In the complete-linkage strategy, q_2 and q_3 will not be in the same group until all the queries are clustered into a single group. But it is apparent that q_3 is a candidate of related queries to q_2 since there is a path from q_2, to q_3 via q_1.

(2)**Weighted-average strategy(WA):**
In the weighted-average strategy, the distance between two clusters is the average distance between all possible pairs of nodes in the two clusters. Its definition is given as

follow:

$$\begin{aligned} d_{hk} &= \frac{\sum_{q_h \in n_h} \sum_{q_k \in n_k} d(q_h, q_k)}{2} \\ &= \frac{\sum_{q_h \in n_h} \sum_{q_i \in n_i} d(q_h, q_i) + \sum_{q_h \in n_h} \sum_{q_j \in n_j} d(q_h, q_j)}{2} \\ &= \tfrac{1}{2} d_{hi} + \tfrac{1}{2} d_{hj} + 0 d_{ij} + 0 |d_{hi} - d_{hj}| \,, \end{aligned}$$

where

$$\alpha_i + \alpha_j + \beta = \frac{1}{2} + \frac{1}{2} + 0 = 1 \quad and \quad \gamma = 0 \,,$$

satisfies the constraint of monotonicity. Since the distance between clusters is calculated as a simple arithmetic average, the weighted-average strategy is computationally easy. When there are unequal numbers of elements in the clusters, the original distances of pairs of queries do not contribute equally to the intermediate calculations, and the final result is therefore said to be weighted. An obvious weakness is that the number of elements in each cluster are not taken into account.

(3)**Group-average strategy(GA):**

Similar to the weighted-average strategy, the group-average clustering also considers the distance between two clusters, defined as ($n_k = n_i + n_j$):

$$\begin{aligned} d_{hk} &= \frac{\sum_{q_h \in n_h} \sum_{q_k \in n_k} d(q_h, q_k)}{n_h n_k} \\ &= \frac{\sum_{q_h \in n_h} \sum_{q_i \in n_i} d(q_h, q_i) + \sum_{q_h \in n_h} \sum_{q_j \in n_j} d(q_h, q_j)}{n_h (n_i + n_j)} \\ &= \frac{n_h n_i}{n_h (n_i + n_j)} \times \left(\frac{\sum_{q_h \in n_h} \sum_{q_i \in n_i} d(q_h, q_i)}{n_h n_i} \right) + \\ &\quad \frac{n_h n_j}{n_h (n_i + n_j)} \left(\frac{\sum_{q_h \in n_h} \sum_{q_j \in n_j} d(q_h, q_j)}{n_h n_j} \right) \\ &= \frac{n_i}{n_i + n_j} d_{hi} + \frac{n_j}{n_i + n_j} d_{hj} \\ &= \frac{n_i}{n_i + n_j} d_{hi} + \frac{n_j}{n_i + n_j} d_{hj} + 0 d_{ij} + 0 |d_{hi} - d_{hj}| \,, \end{aligned}$$

where

$$\alpha_i + \alpha_j + \beta = \frac{n_i}{n_i + n_j} + \frac{n_j}{n_i + n_j} + 0 = 1 \quad and \quad \gamma = 0 \,,$$

satisfies the constraint of monotonicity, thus holding the monotonic property.

The group-average strategy computes the average of distances between all pairs of objects, where each pair is made up of one object from each group, but also consider the size of each cluster. The calculation is slightly more complicated. As a result, each distance contributes equally to the final result, which is therefore said to be unweighted. Its dendrogram is a tree with intense clustering. Furthermore, both weighted- and group-average strategies avoid the pitfalls of the single-link and complete-link criteria, which equate cluster similarity with the similarity of a single pair of queries. One weakness of the group-average strategy is that pairs from the same cluster are not included in the average, i.e., $d_{ij}$when i and j are merged into a cluster k.

TABLE 6.3: Parameters of different HAC strategies

Strategies	α_1	α_2	β	γ
SL	1/2	1/2	0	-1/2
WA	1/2	1/2	0	0
GA	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
α-F	α	α	$1-2\alpha$	0

(4)**Flexible strategy (α-F):**
We can also consider a HAC strategy including self-similarities. The flexible strategy is defined as: $d_{hk} = \alpha d_{hi} + \alpha d_{hj} + (1-2\alpha)d_{ij}$,

which was proposed by Lance et al. [49, 50] as a flexible and monotonic strategy. When α increases from 0 to 1, its hierarchy changes from an almost completely *chained* system into one with increasingly *intense* clustering. A given set of queries may now, by varying the parameters, be made to appear as sharp as a dendrogram a user may desire. Simply speaking, when α is small, e.g., near to zero, the dendrogram of the α-F strategy is similar to the single-linkage strategy while when α is large, e.g., near to one, the dendrogram of α-F strategy is similar to the group-average strategy. If α=0.5, α-F strategy becomes the weighted-average strategy. In experiments, we will show its flexibility to change the dendrograms of clustering, thus producing adaptive recommendation lists for users.

Parameters of the above four strategies in Equation 6.6 are listed in Table 6.3. We will give empirical evidence as to how different HAC strategies affect the quality of recommendation and make use of the flexibility of the α-F strategy to allow users to interactively participate in the recommendation process by changing the value of α.

6.5.2 The Design of Our Ranking Procedure

We provide a sketch of our QUBIC HAC-based ranking procedure in the format of pseudo code in Table 6.4. After a HAC strategy is selected, our algorithm will utilize its result to rank queries in the same group as an input query. Concretely, given n candidate queries, we create an $n \times n$ distance matrix of candidate queries, and apply the chosen HAC strategy on this matrix in three step process (Step 9 $\sim$ Step 11) before forming the recommendation list (Step 12).

First, we create an n-tuple P=(P_1, $\cdots$, P_n) index to compute for each cluster i a nearest neighboring clustering P_i and each P_i for $1 \leq i \leq n$ satisfies the condition that $d(i, P_i)$=min $\{$ $d(i,j)$: $1 \leq i,j \leq n$, $i \neq j$ $\}$. Second, we replace the two clusters i and j by an agglomerated cluster k based on the chosen HAC method (see Step 11.4), and update the matrix D by removing clusters i and j and add the new cluster k. This process continues until no more merge is needed. Although the main weakness of

Table 6.4: Relevance based query ranking and recommendation

Input: an HAC strategy chosen by a user and an input query iq.		
Output: a rank list of related queries to the input query iq.		
9.	If find the the connected component containing the query iq; Else update the graph processing phase to reflect inclusion of iq.	
10.	Choose the H-hop neighbors of the input query as candidates of related queries, and store the distance matrix D($n \times n$ and $n \leq Q$) of candidates (including iq).	O(n^2)
11	Apply the selected HAC strategy on this matrix.	
11.1	Initialize n-tuple P to identify a nearest neighbor of each cluster.	O(n^2)
11.2	For m=n downto 2 do Begin Loop	
11.3	Search D with P to identify a closest pair(i,j) of clusters;	O(m)
11.4	Replace clusters i and j by an agglomerated cluster k; $d_{hk} = \alpha_1 D[h][i] + \alpha_2 D[h][j] + \beta D[i][j] + \gamma \lvert D[h][i] - D[h][j] \rvert$	O(1)
11.5	Update D to reflect deletion of i and j and compute distances; between k and all remaining clusters(h)	O(m)
11.6	Update P to reflect deletion of i and j and inclusion of k; End Loop	O(m^2)
12.	The successive merging operations of HAC naturally form an ordered list.	
12.1	$M[iq]$: the first merging distance for the input query iq;	O(n)
12.2	For cq=1 to n ($cq \neq iq$) Being Loop	
12.3	$M[cq]$: the first merging distance for the candidate query cq;	O(n)
12.4	$M[(iq, cq)]$: the merging distance for the cluster that include cq; and iq	O(n)
12.5	$R[cq] = \lvert M[iq] - M[(iq, cq)] \rvert + \lvert M[cq] - M[(iq, cq)] \rvert$; End Loop	O(1)
12.6	Quicksort $R[n]$ BY ASCENT;	O($nlogn$)
12.7	Return the top similar queries but delete the candidate queries whose $D[iq][cq]$ are smaller than 0.2;	O(n)
13.	If the user is unsatisfied with the list, selects another HAC strategy; Go to 11.1; Else end this searching process.	

agglomerative clustering methods is that they do not scale well with time complexity of at least O(n^2), the complexity of the clustering and ranking process depends on the number of candidate queries obtained in Step 10 which is smaller than the total number of queries and is typically dependent on the number of hops used in traversing the query affinity graph. If the users want more diverse queries and increase the number of hops, it will take more time to traverse the affinity graph and to cluster and rank them.

The process of the HAC clustering is stored as an n by 2 matrix M which contains the combination distances between merging clusters at the successive stages. Let n be the number of candidates. Row i of the matrix describes the merging of clusters at the step i of the clustering. If a number j in the row is negative, then the single page

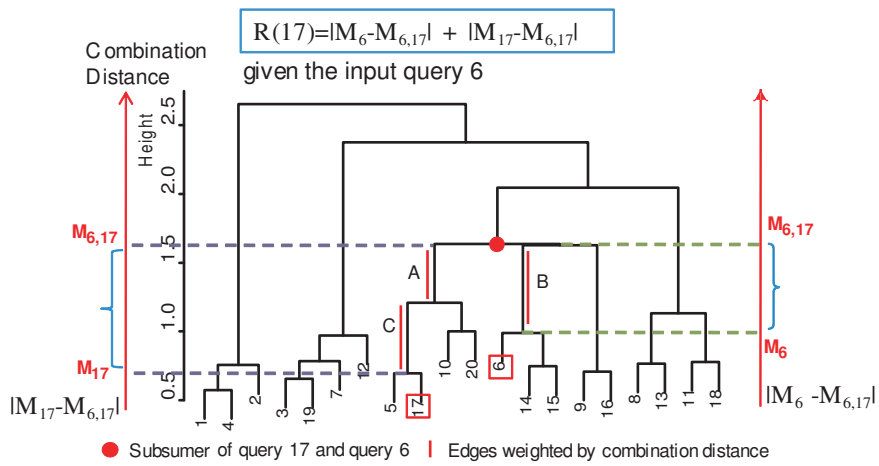

FIGURE 6.4: HAC based ranking using tree distance

j is merged at this stage. If j is positive, the merger is with the cluster formed at stage j of the algorithm. We show that the successive merging operations of the HAC algorithm naturally form an ordered list. A HAC strategy builds a hierarchical structure (dendrogram) where an input query is merged with a group(query) at a distance ($M[iq]$) in the first time (Step 12.1), and the first merging for a candidate query is at a distance ($M[cq]$) (Step 12.4). In addition, the input query (iq) and the candidate query (cq) will come together at a combination distance ($M[(iq, cq)]$) (Step 12.5). Therefore, each of the three steps Step 12.1, 12.4, and 12.5 requires $O(n)$ searching the matrix. Then, the distance score between the two queries is estimated to$|M[iq] - M[(iq, cq)]| + |M[cq] - M[(iq, cq)]|$ which ranks each candidate (Step 12.6).

We use a concrete example to explain how HAC-based ranking works. A clustering structure produced by our experiments is illustrated in Figure 6.4 where "Height", the label of the vertical axis, means the combination distance at each merging operation. As shown in Figure 6.4, the dendrogram shows the relationships of queries arranged like the branches of a tree. Computing the distance (similarity) of two queries in the dendrogram can utilize the tree-distance based ranking mechanism which usually counts the number of edges between two nodes in a tree. In our HAC dendrogram, the edges are weighted by the combination distances of clustering, e.g., A, B, and C edges in Figure 6.4. Therefore, our HAC-based ranking computes the sum of weights of edges between two queries (nodes). For example, given query 6 as the input query, the subsumber between query 6 and query 17 is marked as a dot in Figure 6.4 and this subsumber is produced at the combination distance $M_{6,17}$. The sum of weights of edges between query 17 and the subsumber is $|M_{6,17} - M_{17}|$ (leftmost) and the sum of weights of edges between query 6 and the subsumber is $|M_{6,17} - M_6|$ (rightmost). Then, the distance between query 6 and query 17 is $|M_{6,17} - M_{17}| + |M_{6,17} - M_6|$ as shown in the top of Figure 6.4. We empirically show this HAC-based ranking is effective in Section 6.6.

A naïve ranking is simply based on the Jaccard, L_1 or Cosine similarity measures. Compared with our ranking methods, the naïve ranking only considers the similarity between two nodes without similarity propagation. Give a simple example shown in Figure 6.5 which is the dendrograms of α-F strategy (α=0.02 and 0.5) applied on Figure 6.3. The

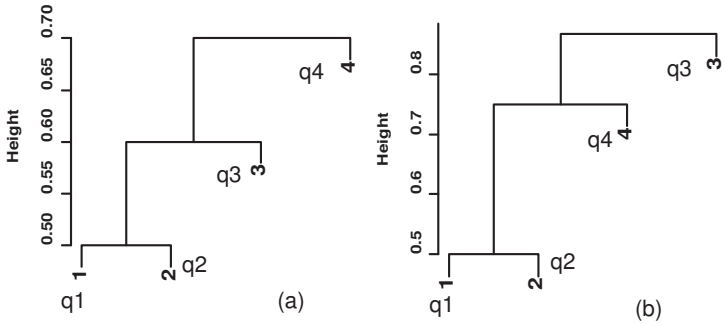

FIGURE 6.5: (a) $\alpha = 0.02$ and (b) $\alpha = 0.5$ in Equation 6.6

similarity scores of pairs of queries are $sim(q1, q2) = 1 - 0.5 = 0.5$, $sim(b, c) = 0$, $sim(a, c) = 1 - 0.6 = 0.4$, $sim(d, a) = 1 - 0.7 = 0.3$, and $sim(d, b) = 1 - 0.8 = 0.2$, as illustrated in Figure 6.3. In Figure 6.5(a) the query $q1$ and the query $q2$ are merged in the first combination under $\alpha = 0.02$. The query $q3$ will be merged with $q1(q2)$ in the second combination, and then the query $q4$ will be merged in the third combination. The ordering of related pages of the query $q2$ is $q1$, $q3$, and $q4$, despite of $sim(q2, q3) = 0$ and $sim(q2, q4) = 0.2$. Therefore, the tree distance based ranking naturally completes the similarity propagation from $q3$ to $q2$ via $q1$. Moreover, if the user chooses $\alpha = 0.5$ in Figure 6.5(b), the ordering list becomes $q1$, $q4$, and $q3$. Our HAC-base ranking is able to not only capture the similarity propagation between queries, but also adaptively output the ordering list of related queries. Users will be supplied with more candidates of related queries. More experiments results are reported in Section 6.6.

6.5.3 Discussions

As we discussed earlier, our similarity measure between queries is based on the common URLs returned from submitting both queries to a search engine (e.g., the number of common URLs in Equation 6.1 and the weighted common URLs in Equation 6.3 and 6.5). If these related queries are identical or give an identical result to the original query, then they add no value to the query recommendation quality and effectiveness, and thus cannot improve the level of satisfaction of the users. Naturally, we want to suggest queries that achieve comparable search results.

One way to approach this goal is to remove the candidates whose distance scores are smaller than a threshold.(e.g., in Equation 6.1 in step 12.7, the value of 0.2 means that

TABLE 6.5: Selection of the number of top search results

#	top 10	top 20	top 30	top 40	top 50
0	272301(85%)	241310(76%)	207593(65%)	177225(55%)	147111(46%)
1	34936(11%)	58553(18%)	79903(25%)	94533(30%)	105327(33%)
≥ 2	1823(0.6%)	9196(2.9%)	21564(6.7%)	37301(12%)	56622(18%)

the overlap of common URLs exceeds 80%). This ensures that our approach does not try to recommend *exactly similar* queries, but *sub-similar* ones. From this viewpoint, query recommendation is different from query clustering. The latter is to mine the exactly similar queries, but the former is to find sub-similar ones. It is worth noting that in the first prototype of QUBIC we simply choose 0.2 as a filtering threshold, in real application, users can get more similar queries by decreasing the value or obtain less similar ones by increasing it. The reason that we did not apply such filtering in Step 5 of the second phase is because these candidates can work as bridge to propagate similarity between queries during the HAC clustering.

6.6 Experiments

In order to test the effectiveness of the QUBIC system for query recommendation, several experiments are conducted. Firstly, we test the precision quality of query recommendation to evaluate the number of related queries in a recommendation list. Then, we introduce Entropy as an evaluation measure to reflect the quality of individual lists in terms of homogeneity of related queries in a recommendation list. Finally, We also provide a case study to illustrate how our system can adaptively output recommendation lists by utilizing the flexibility of the HAC strategy.

6.6.1 Data Sets

In the experiments, we used two data sets: QueryKDD and QueryTREC for evaluation. The former is the 800 queries of KDD cup 2005 data set [1] that are labelled by three people. The latter is the "10k" stream of the Efficiency task topics of the 2006 TREC Terabyte Track [2] (this task contains 6 topic streams: "all": full 100k topics, "stream-1" through "stream -4": four concurrent streams, and "10k": a small stream for the comparative efficiency task). Given each query, the top 50 URLs of search results were offered by Google API [3]. We preprocessed the URLs and queries by mapping their characters to lowercase and by chopping each retrieved URL into its hostname.

[1] http://www.acm.org/sigs/sigkdd/kddcup/index.php
[2] http://trec.nist.gov/data/terabyte06.html
[3] http://code.google.com/apis/soapsearch/

TABLE 6.6: Query-URL bipartite graph

Data Set	Queries	URLs	Edges
QueryKDD	800	26,206	39,599
QueryTREC	10,000	147,761	491,956

TABLE 6.7: Affinity graph of queries

QueryKDD data set			
	Queries	Edges	Neighbors/Query
ALL	695	18,762	27
Jaccard	678	8,841	13
Cosine	683	9,511	14
L1 norm	684	10,149	15
QueryTREC data set			
	Queries	Edges	Neighbors/Query
ALL	9559	1,751,148	183
Jaccard	1297	6,738	5
Cosine	2355	7,600	3
L1 norm	1275	3,156	2.5

6.6.2 Statistics about Data Sets

We did a preliminary analysis of the number of top URLs used in our scheme on the QueryKDD data set, as shown in Table 6.5. The first column (#) represents the number of common URLs between queries, and the first row (top 10 $\sim$ top 50) represents the number of pairs of queries whose total number reaches $(X * X - X)/2$ (here X is set to 800). From Table 6.5, we observed that about 85% of the pairs of queries had no common URLs in the top 10 of the search results, while in the top 50 search results, the number of pairs of queries with no common URLs was reduced to 46%. The distance (similarity) measures discussed in Section 6.4 rely on the common URLs shared by two queries. If the number of returned search results is too small to get the URL overlap between two queries, we have to increase the number of returned results. Therefore, in the following experiments we stored the top 50 of search results for each query.

Table 6.6 and Table 6.7 summarize the statistics of the query-URL bipartite graphs and the affinity graphs of queries generated from our two data sets. Note that the column "URLs" means the number of distinct URLs. Because we chopped the original URLs into their hostnames, it is possible that there are duplicate hostnames in the top 50 search results.

In Table 6.7, the row "ALL" represents that the affinity graphs of queries are set up with the threshold $\delta = 1$ (Step 5 in Table 6.2). For each query, we sorted the other queries in terms of distance scores. We found that the scores of the bottom queries (from 100th or

200th) are larger than 0.98 in QueryKDD or 0.85 in QueryTREC. Moreover, our task is not to find all the similar queries like query clustering, but to recommend the more relevant ones. Therefore, the values of the threshold δ for QueryKDD and QueryTREC are 0.98 and 0.85 respectively.

One observation from Table 6.7 is that the number of queries in the affinity graph is smaller than that in Query-URL bipartite graph (695 vs. 800 and 9559 vs. 10,000) because some queries are isolated. "isolated" means that a query has no common URLs with any other queries. The column "Neighbors/Query" gives the single hop neighbors per query. To include more candidates during recommendation, users can set H $\geq$ 2 hops from any input query (Step 10 in Table 6.4). With increasing the value of H, the HAC clustering will become slowly. We computed the number of candidates of each query according to different values of H. For QueryKDD dataset, when H=4, the number of queries whose total number of candidates is larger than 700 is 702 (the total number of queries is 800); when H=2, there is only one query whose number of candidates reaches 571. For QueryTREC dataset, if H=4, most of queries have the number of candidates larger than 8000; while H=2, most of them have the number of candidates smaller than 2000. Therefore, for evaluation, we choose H=3 to keep the overall data within a reasonable size. We observed that the produced affinity graph is an unconnected graph which may be subdivided into connected components. This fact encouraged us to extract such connected components in advance (Step 6 $\sim$ Step 7 in Table 6.2), thus reducing the computation complexity in the HAC-based ranking phase.

6.6.3 Evaluation Measures

6.6.3.1 Precision

The precision (P) at the top N queries of a recommendation list is defined as:

$$P@N = \frac{Related\ Queries}{N}. \tag{6.7}$$

The measure $P@N$ means how many valuable answers our algorithm gives at the top of recommendation lists. We set N=10 in the following evaluations. Query recommendation is a ranking problem. We can evaluate our system by other evaluation measures like MAP and $NDCG$ [62] which are also used for ranking problems, especially in Web search. Because the length of a term based query is much shorter than that of a Web page, Web users can read the top ten or twenty recommended queries much more quickly than they browse and click the top Web pages one by one. In the evaluation of query

recommendation, we are more interested in the number of related queries in a recommendation list, i.e., *Precision* than the order in which the queries are returned such as MAP, $NDCG$ and so on.

6.6.3.2 Entropy

In the queryKDD data set the manual categorization information of queries are available. Each query is followed by its top five categories labeled by three human experts. There may be fewer than five categories for some of the queries and these categories come from 67 predefined categories. The category information assumes that it is possible to perform a direct comparison of the recommendation output. Our hypothesis is that an optimal recommendation output consists of queries with the same class labels. We describe a quantitative evaluation measure based on this criteria. An information entropy approach can evaluate the quality of a set of clusters according to the original class labels of the data [8]. By replacing *cluster* with *ranking list*, we compute the entropy of each ranking list of query recommendation, defined as

$$Er_i = -\sum_j \frac{n(l_j, r_i)}{n(r_i)} log \frac{n(l_j, r_i)}{n(r_i)}\,, \tag{6.8}$$

where $n(l_j, r_i)$ is the number of the related queries in the ranking list r_i with a predefined label l_j and $n(r_i) = \sum_j n(l_j, r_i)$ is the number of the related queries in the ranking list r_i. The overall *ranking list entropy* Er is then given by a weighted sum of individual cluster entropies by

$$Er = \frac{1}{\sum_i n(r_i)} \sum_i n(r_i) Er_i\,. \tag{6.9}$$

The entropy of ranking lists reflects the quality of individual lists in terms of homogeneity of related queries in a recommendation (a smaller value indicates a higher homogeneity).

6.6.4 Results and Discussions

6.6.4.1 Experiments on the QueryKDD Data Set

Precision

For precision evaluation, if two queries share at least a same category, we regard them as related queries. The experimental results averaged by three labellers are shown in Figure 6.6. The single-linkage strategy performs worst among all the HAC-based rank mechanisms. It is even worse than the naïve ranking and the performance is lowered by 3.9%, 11.3%, and 1.9% for the three similarity measures. The reason is that the

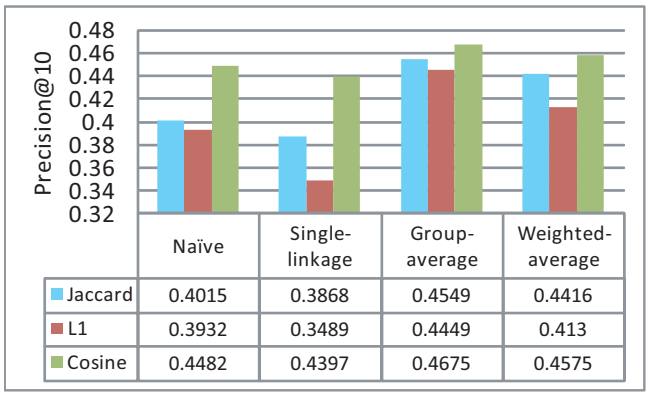

FIGURE 6.6: Precision@10 of QueryKDD data set

single-linkage strategy is sensitive to outliers since it merges in each step the two clusters whose two closest members have the smallest distance or the two clusters with the smallest minimum pairwise distance. While the group-average and weighted-average strategies got higher precision than that of the naive one under all the three measures. The most improvement is about 13.3% when using the *Jaccard* measure and the group-average strategy. The two strategies are robust to outliers because they consider the distance between one cluster and another cluster to be equal to the average distance (or Equation 6.6) from any member of one cluster to any member of the other cluster. Furthermore, the group-average strategy produces higher precision scores than the weighted-average strategy because the former also considers the size of clusters in its cluster processing.

Among the three similarity measures, The *Cosine* measure is the best one and the L_1 norm is worst, while the *Jaccard* measure shows better than the L_1 norm. The *Jaccard* measure simply computes the number of common URLs between queries, thus assigning equal weights to all the URLs in a query. While the L_1 measure weight a URL by its frequency in the search result of a query. As we know, popular sites like Google and Yahoo! may be returned within the search results of a query, which will be noisy irrespective of the content of the query. Since the L_1 norm may assign more weights on these URLs, this weighting function hurts the precision in our context. The other weakness of the *Jaccard* measure is that it does not distinguish between two queries which have the same one URL and two queries which have the same two URLs. The URL-vector based $tf * idf$ weighing method is more effective than both of the *Jaccard* measure and the L_1 norm since it assigns less weights on URLs with higher query frequency that might be less informative to represent a distinct query, and more weights on URLs with higher occurrences in a query. On average, the group-average strategy with the *Cosine* measure is the winning combination in terms of precision@10.

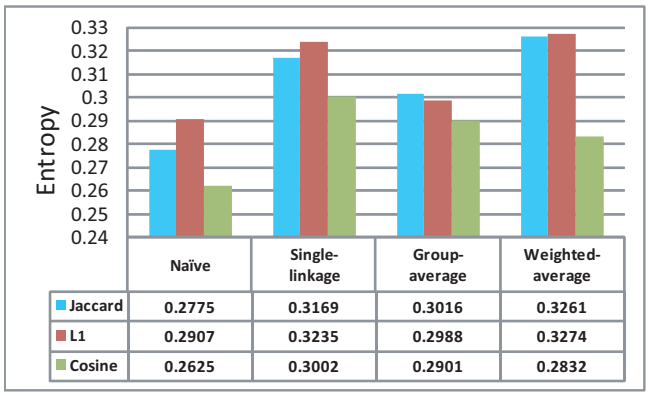

FIGURE 6.7: Entropy of ranking lists of QueryKDD data set

Entropy

Entropy evaluation defined in Equation 6.9 is illustrated in Figure 6.7. With regard to the overall performance, Figure 6.7 shows that the group-average strategy with the *Cosine* measure produced lowest entropy values among the three HAC based rank mechanism (smaller is better for entropy). But the HAC based ranking methods show higher entropy that the naïve ranking, which does not beyond what we expected. Because the naive rank mechanism directly uses the similarity measures, it has lower possibility to include noisy queries. However, the HAC strategies try to get more relevant queries by similarity propagation at the same time they has higher risk to meet noisy queries. Therefore, the final lists of the naïve one are purer than those of HAC strategies. According to precision, entropy and robustness, the group-average strategy with the *Cosine* measure is the optimal combination. These results prove that our QUBIC approach is effective in query recommendation.

6.6.4.2 Experiments on the QueryTREC Data Set

Precision

Unlike the QueryKDD data set, the QueryTREC data set does not supply us with the categorization information. Thus, it is not feasible to use the entropy of rank lists as our evaluation metric. Three human evaluators are invited to judge the performance of our algorithm. After running our three-phase approach with group-average strategy, we can recommend a list of related queries for each query. We then randomly selected 30 queries as our test data. The three evaluators worked separately without knowing how our algorithm works. *Related Query* in Equation 6.6 is the number of manually tagged related queries. Their averaged results are reported in Table 6.8. It is clear that the evaluation approach is somewhat subjective. However, the concept of similarity is also subjective due to the various information needs of users. It is not easy to construct an objective test data set for our problem at the current stage.

TABLE 6.8: Precision@10 of QueryTREC data set

	Naïve	Group-average
Jaccard	0.43	0.56
L_1 norm	0.32	0.50
Cosine	0.64	0.73

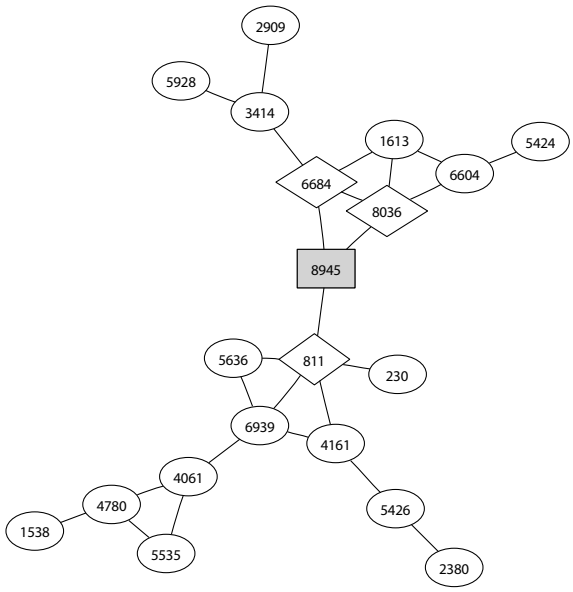

FIGURE 6.8: Affinity graph of the query with the topic number= 8945

The group-average strategy generally shows the best performance among all HAC strategies. Therefore, in the QueryTREC, we compare the group-average strategy with the naïve ranking. The evaluation results are reported in Table 6.8. From the experiment results of the QueryKDD data set, the *Cosine* measure achieves best performances among all the three similarity measures because the *Jaccard* measure overlooks the frequency information of URLs and the L_1 measure cannot avoid the side-effect of popular sites on the quality of query recommendation. In addition, the group-average strategy still shows higher precision scores than the naïve ranking. These results are consistent with the results of QueryKDD data set. We can conclude that the *Cosine* measure using the conventional TFIDF term weights of documents is effective to represent the queries in a URL-vector space model, and HAC-based ranking is more effective for query recommendation than the naïve ranking.

Case study

In our case study, we discuss the top 10 related queries returned by the *Cosine* measure for the query "state of wa department of social and health service division of child support" (topic number= 8945 in the "10k" stream of the Efficiency task topics of the 2006 TREC Terabyte Track). The affinity graph of this query is shown in Figure 6.8. A tradi-

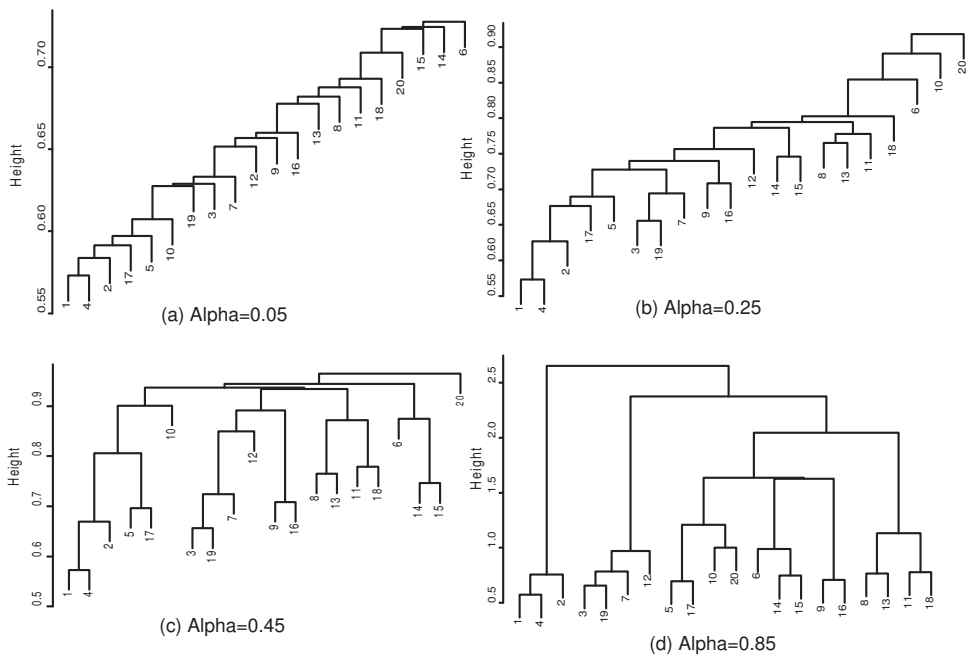

FIGURE 6.9: Effect of varying HAC strategies. Measure: Cosine; Height: the combination distance at each merging.

tional method of finding related queries is to select queries with higher similarity scores computed by a measure, such as *Jaccard*, *Cosine*, and so forth. Such queries are only adjacent vertices of a targeted query (e.g., vertices 811, 6684, and 8036 in Figure 6.8). As we have discussed, however, similarity propagates from query to query. From the graph theoretical viewpoint, it means that related queries refer to queries from which there are paths to the targeted query (e.g., vertices 1613, 4061, and 6604 in Figure 6.8).

Our HAC strategy can identify unadjacent related queries through propagating similarity over other related ones, as listed in Table 6.9. Moreover, in Figure 6.9, as α increases from 0.05 to 0.85 in the α-F strategy, the hierarchy changes from an almost completely "chained" system to one with increasingly intense clustering. Using query 1 as the input query and query 5 in Figure 6.9(d) as a concrete example, we know $M[1]= 0.575$, $M[5]$=0.7, and $M[(1,5)]$=0.8. Then, the distance between query 1 and query 5 ($R[5]$) is 0.375 which may vary with hierarchical structures produced by different values of α in the α-F strategies. We have listed the recommendation results for a query in Table 6.9 using different HAC strategies.

It is obvious that the order in the list of related pages partly changes as well. We represented this change in Table 6.9 where "8945 : 1" means that "8945" is the topic number in QueryTREC data set and "1" is the query number in Figure 6.9. In our adaptive scheme, if a user inputs a query to retrieve web pages, our system will supply her with a list of related queries from our data set. Then, the user can formulate his/her query and do a new search, or she may also ask our system to produce a new list of

TABLE 6.9: Recommendation lists of the query "8945"

Rank	α=0.05	α=0.25	α=0.45	α=0.85
0	8945 : 1	8945 : 1	8945 : 1	8945 : 1
1	**6684 : 4**	**6684 : 4**	**6684 : 4**	**6684 : 4**
2	**8036 : 2**	**8036 : 2**	**8036 : 2**	**8036 : 2**
3	1613 : 17	1613 : 17	5424 : 10	5424 : 10
4	6604 : 5	6604 : 5	6604 : 5	230 : 20
5	5424 : 10	4161 : 12	1613 : 17	5928 : 6
6	**811 : 19**	5636 : 7	230 : 20	4161 : 12
7	6939 : 3	5426 : 9	5928 : 6	5636 : 7
8	5636 : 7	2380 : 16	4161 : 12	4780 : 11
9	4161 : 12	6939 : 3	4780 : 11	1538 : 18
10	5426 : 9	**811 : 19**	1538 : 18	**811 : 19**

related queries by changing the value of α, e.g., adjusting a sidebar. As we discussed in Section 6.5, when α is near to zero, the dendrogram of the α-F strategy is similar to the single-linkage strategy; when α is near to one, its dendrogram is similar to the group-average strategy. If α=0.5, α-F strategy becomes the weighted-average strategy. This can guide users how to adaptively get recommendation lists. The naïve ranking cannot supply users with such diverse lists of query recommendation due to its locality and lack of effective similarity propagation.

6.7 Summary

We presented QBUIC, a pseudo relevance based query suggestion approach [55]. Our query recommendation system can adaptively recommend related queries to a given query by analyzing the query-URL history, consisting of three phases, i.e, preparation, graph generating, and HAC-based ranking phases. In the first phase, a query-URL bipartite graph was build by retrieving search results of queries from a search engine. In the second phase we generated the query affinity graph using the query-URL vector based similarity measures, and extracted connected components from the query affinity graph. In the third phase, we employed a selected HAC strategy to output recommendation lists of related queries. Experimental results are very encouraging: many similar queries are grouped, which enables users to pick up different recommendation lists of related queries through the adjusting of the parameter α in the flexible strategy. Moreover, the group-average strategy based ranking with the *Cosine* measure shows the highest precision in two data sets. These experimental results demonstrate the effectiveness of our approach.

Chapter 7

Suggesting Related Keyword based Queries Using Hidden Topic Model

Through the studies in Chapter 5 and Chapter 6 we have reached a consensus that the keywords in defining a query alone are not rich enough to achieve high suggestion accuracy. We have studied alternative feature spaces extracted from implicit and pseudo relevances which, however, have inherent defects for realtime Web-scale applications. In this chapter, we propose a novel two-step approach to solve the problem by building a generative probabilistic model.

The rest of the chapter is organized as follows. We first introduce our research motivation in Section 7.1. Section 7.2 presents a framework of our approach. Then, we briefly introduce the proposed hidden topic model in Section 7.3. Section 7.4 explains the proposed two-step approach in detail. We report an empirical study in Section 7.5. Finally, the chapter is concluded is outlined in Section 7.6.

7.1 Introduction

We categorize the research efforts on related query suggestion into the following three classes of strategies according to features used in query similarity computation:

1) Query term based suggestion is easily applicable, which intuitively thinks the candidate queries having common words with a current input query to be related. However, the very small word overlap between short Web queries makes it hard to accurately

estimate their similarity. One reason is probably that queries phrased in the same list of keywords may be meant for different results by different users (i.e. polysemy) , e.g., apple is related to fruit or computer. The other reason is that queries can be phrased with different keywords but are meant for the similar information need (i.e. synonymy), e.g., car and automobile. Therefore, many current studies focus on how to improve the performance of this naïve suggestion strategy.

2) Pseudo relevance based suggestion induces the similarity between the two queries by enriching query expressions with additional features from the top results returned by search engines [33, 55, 101]. The pseudo relevance, however, contains no evidence of user judgments [62].

3) Implicit relevance based suggestion extracts additional features from the clicked search results of Web logs to calculate query-to-query similarity [2, 3, 5, 52, 61, 63, 97, 101]. It is more useful than pseudo relevance since the click choice made by a large number of users does suggest a certain level of relevance from the viewing point of actual users.

The three kinds of strategies have their own advantages and disadvantages. At first glance, though implicit relevance based suggestion is the more practical than other twos since it reflects the real opinions of web users, it has a critical drawback. If the input query by a user is totally new to current Web logs, its click information is not accessible from the query achieve for query similarity computation. In this case, we may think of using pseudo relevance to expand the expressive attributes, which, however, is usually relied on downloading the search results returned by search engines. This makes it difficult to be deployed under the circumstances of operational restriction where the cost of utilizing the retrieved search results is extremely high for large volumes of queries. Although the query term based suggestion strategy is the simplest and does not require any external sources, its suggestion results are always unsatisfactory. In summary, an important and crucial research issue of query suggestion is how to automatically and effectively suggest the related Web queries of target queries, especially when they are in short words and are not stored in Web logs. In this chapter, we aim to address the above question.

Bearing this problem in mind, we propose a novel two-step approach to query suggestion in this chapter. In the offline model-learning step, we collect an appropriate training data by gaining external knowledge from Web to learn a generative probabilistic model for Web queries. The training data is carefully selected by considering the topic consistency with queries. Then, employing an iterative estimation procedure results in discovering

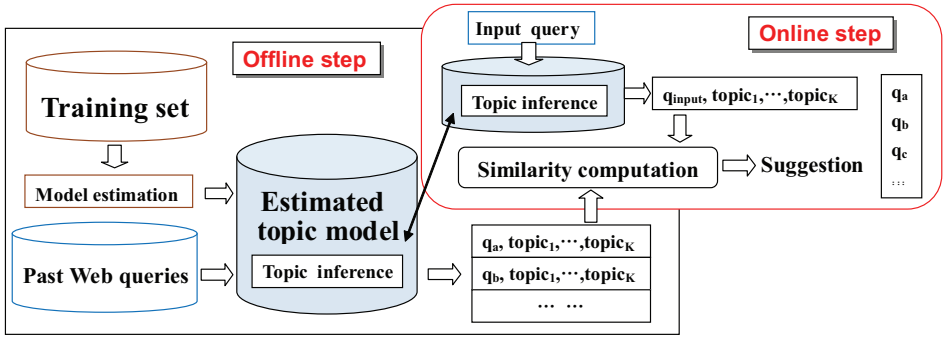

FIGURE 7.1: The framework of our approach

a latent topic model, which governs the co-occurrence of web queries. With this model, for each candidate query, its posterior distribution of hidden topics is determined. In the online query suggestion step, the topic inference is done for an input query. We calculate the similarity between the input query and candidate queries based on their corresponding topic distributions. Finally, a suggestion list of candidate queries is produced according to the similarity scores, which gives the hints to search engine users in rephrasing their query formulation.

We make the following contributions. First, the proposed probabilistic topic model allows us to represent Web queries at a latent semantic level rather than by lexical occurrence alone. Second, the advantage of topic inference procedure used in the proposed model overcomes the limitation of queries having to be presenting in Web logs, which often suffers other query suggestion methods. And it is not needed to extract the external sources from Web in the online step, which makes our approach more adaptive to be used in demanding operational environments, such as large-scale Web search services. Third, the experimental results conducted on the KDD-cup 2005 data clearly indicate that our approach outperforms two baseline methods in terms of suggestion precision. Furthermore, we empirically investigate the effect of the number selection of hidden topics on suggestion quality.

7.2 Framework of Our Approach

Suppose that we have already obtained a set of candidate queries which are past queries submitted by previous users. Our task is to suggest the most related candidates for an target input query. The overall framework of our approach as shown in Figure 7.1 consists of two steps.

In the offline model-learning step, we first choose a suitable training data which should be rich enough to cover as broad and diverse as possible words, concepts, and topics

that are relevant to Web queries. Then, model estimation applies the topic analysis on the training data, which can be performed by using one of the well-known hidden topic analysis models such as probabilistic latent semantic analysis (pLSA) [38] or Latent Dirichlet Allocation (LDA) [7]. Here we chose LDA because this model has a better defined generative model of texts than pLSA and an advantageous capability of automatically conducting topic probability assignments for a previously unseen text. LDA will be briefly introduced in Section 7.3. With LDA model, we are able to estimate the latent topic model for candidate queries. In the online suggestion step, we also use the same trained topic model to infer the topic distribution of an target input query. Based on the inferred topic distribution of the query, the query in an original expression of lexical occurrence is transformed into a concise topic space, which captures the semantic meaning of the query. We can use the transformed query expression to measure the similarity between various queries. After the similarity score between the input query and each of candidate queries is calculated based on their topic distributions, our approach suggests the most related queries with the highest similarity scores to the current users from the candidates.

Last, we would like to note that additional cost might be incurred when the terms of an input query are not in the training set, since we need to collect and add its relevant information to the training data, re-learn the topic model. One of our ongoing research is to devise incremental optimization techniques to reduce the cost of updating the topic model.

7.3 LDA Model

LDA is a probabilistic generative model of a text corpus. The basic idea is that documents are represented as random mixtures over latent topics and each topic is characterized by a distribution over words. More theoretically, LDA is on a basis of an assumption that there exists an unseen structure of "topics" or "themes" in the text corpus, which governs the co-occurrence observations. As such, the intuition behind LDA is to discover this latent topic structure via extracting the abstracts of the original co-occurrence activities. The generative procedure of LDA model is shown in Figure 7.2 and the notations used are described in Table 7.1.

In LDA, a document $d_m = \{w_{mn}, n = 1, ..., N_m\}$ is generated by picking a distribution over the topics from a Dirichlet distribution ($Dir(\alpha)$). And given the topic distribution, we pick the topic assignment of each specific word. Then the topic assignment for each word placeholder $[m, n]$ is calculated by sampling a particular topic $z_{m,n}$ from the multinominal distribution of $Mult(\theta_m)$. And finally, a particular word of $w_{m,n}$

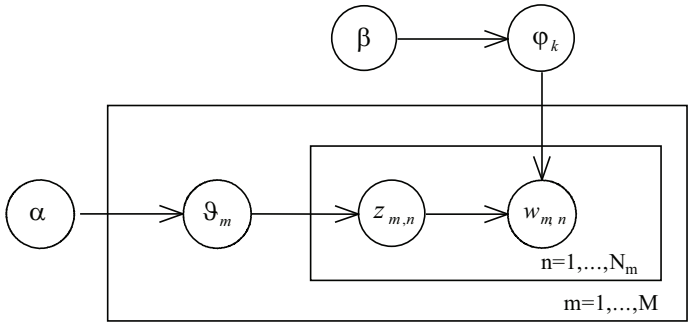

FIGURE 7.2: The illustration of LDA

TABLE 7.1: Notations of LDA model

M:	the number of documents
K:	the number of topics
V:	the size of vocabulary
α, β:	Dirichlet parameters
θ_m:	the topic assignment of the document m
Θ:	the topic estimations of the corpus, a $M \times K$ matrix
φ_k:	the word distribution of the topic k
Φ:	the word assignments of the topics, a $K \times V$ matrix

is generated for the placeholder $[m, n]$ by sampling its weight from the multinomial distribution of $Mult(\varphi_{z_{m,n}})$. Known from the above description, we can formulate a joint distribution of a document d_m, a topic mixture of d_m, i.e. θ_m, and a set of N_m topics, i.e. $z_{m,n}$, defined as

$$P_r(d_m|\alpha, \beta) = \int P_r(\theta|\alpha) \prod_{n=1}^{N_m} P_r(w_{m,n}\,|\,\varphi_{z_{m,n}}) P_r(z_{m,n}\,|\,\theta_m) d\theta_m\,, \tag{7.1}$$

where α and β are Dirichlet parameters. Under this model, documents can be associated with multiple topics.

7.4 Our Two-step Approarch

We dedicate this section to describe our query suggestion approach in detail.

7.4.1 Offline Model-learning Step

7.4.1.1 Collecting A Training Dataset

Our idea is to build a latent topic model for Web queries. It is vital to find a suitable data collection for model training. Since search engine users submit keyword-based

queries to satisfy their information needs from Web, it is necessary to obtain a query-term dataset at first for the purpose of model training. The query, however, is always in a representation of short words, thus it is not applicable to utilize the terms included in the query alone for training. Like the feature expansion methods used in text mining, here we intend to refer to external data sources to enrich the expressive features. There are a number of potential data sources usable as training data , such as Wikipedia, Open Directory Project (ODP), Web archive crawled before, Web data in TREC Web track, and so on. In other words, we think that the meanings of Web queries are encoded in such Web corpora. For example, a log file of a search engine stores a large-scale Web archive which can be used as a potential training set to satisfy the diversity and the coverage of queries. Section 7.5 discusses an implementation on how to construct a training dataset using Google APIs based on a collection of queries on which we will conduct evaluation. Once the training dataset is completed in a representation of document-term array where each row denotes a document and each column corresponds to a term, we can then employ the estimation procedure of LDA model to derive the latent topic model from the training dataset.

7.4.1.2 Model Estimation

In order to use the topic model obtained by LDA, we need to solve the problem of computing the posterior distribution of the hidden topics given a query. In fact we think that the query is a specially short document. Estimating parameters for LDA by directly and exactly maximizing the likelihood of the whole data collection in Equation 7.1 is intractable. The solution to this is to use various alternative approximate estimation methods. Here we employ the variational EM algorithm [7] to find the variational parameters that maximize the total likelihood of the corpus with respect to the model parameters of α and β:

$$(\alpha_{est}, \beta_{est}) = \max l(\alpha, \beta) = \max \sum_{m=1}^{M} \log Pr(d_m \mid \alpha, \beta)\,. \tag{7.2}$$

The variational EM algorithm is briefly described as follows:
1) (E-step) For each document, find the optimizing values of variational parameters θ_m^* and φ_m^*.
2) (M-step) Maximize the resulting low bound on the likelihood with respect to model parameters α and β. This corresponds to finding maximum likelihood estimates with the approximate posterior which is computed in the E-step.
The E-step and M-step are executed iteratively until a maximum likelihood value reaches. Meanwhile, the calculated estimation parameters can be used to infer topic distribution

of a new document by performing the variational inference. More details with respect to the variational EM algorithm are referred to [7].

7.4.1.3 Topic Inference

After doing topic estimation for the training dataset, we eventually obtain the posterior parameters of the LDA model $\alpha$and β as well as the topic assignments of documents and terms. Then the trained topic model could be used for topic inference for a target input document (or a query in the form of term vector). In this case, the inference is done through the variational algorithm [7]. As a result, an inferred topic distribution of the target query is calculated, which reflects the likelihood of various topic assignments of the query. As the dimensionality of the topic space is much smaller than that of the original term space, the LDA operation could be viewed as a dimensionality reduction method where the latent semantic topic of the document is optimally approximated. In other words, after LDA model estimation, the expression format of the input query is transformed from a high-dimensional sparse term space to a low-dimensional latent topic space.

7.4.2 Online Suggestion Step

After completing the offline step consisting of model estimation and topic inference, each input query is re-expressed by its topic distribution in addition to its intuitive term expression. Then, based on the transformed expression, we can measure the mutual similarity between various queries. Here we calculate two kinds of similarity measures: one is the similarity of topic distribution alone, and the other is the similarity of combined term-topic vector. As discussed above, each input query is expressed as a probability distribution over the latent topics (θ_{ij}), by selecting a threshold value μ, we can obtain the following vector expression with respect to significant topic distribution alone: $q_i = q_{tp} = \{s_{i1}, ..., s_{ik}, ..., s_{iK}\}$, where

$$s_{ij} = \begin{cases} 0 & : \quad \theta_{ij} < \mu \\ \theta_{ij} & : \quad otherwise \end{cases} .$$

The parameter μ depends on the topic distribution learned from the training data. In our experiments, we set μ to be 0.1. Furthermore, we can also formulate a combined expression of the input query by integrating the topic assignment with the term presence: $q_{tm,i} = \{t_{i1}, ..., t_{ij}, ..., t_{i,V}\}$, where t_{ij} is the term frequency of the jth term in the ith query:

$$q_i = q_{tp,i} \cup q_{tm,i} = \{s_{i1}, ..., s_{ik}, ..., s_{iK}, t_{i1}, ..., t_{ij}, ..., t_{iV}\} .$$

The similarity function used is the well-known cosine coefficient, defined as:

$$sim(q_i, q_j) = \cos(q_i, q_j) = \frac{(q_i \cdot q_j)}{\|q_i\|\|q_j\|} .$$

The cosine function is widely employed in information retrieval [62].

Finally, our approach outputs a ranked list of candidate queries according to their similarity scores with the input query. We will evaluation the suggestion performance using the two kinds of query expressions. Notice that in the online suggestion step we do not require any external sources while pseudo relevance based suggestion has to crawl and process relevant information from search engines and implicit relevance based suggestion assumes that the current input query is already in a set of past queries.

7.5 Experiments

In this section, we evaluate the suggestion precision of our approach compared with two baselines.

7.5.1 Experimental Setup

The query data set used are the 800 categorized queries provided by KDD cup 2005 [1] for the reason of easily and accurately evaluating. Given the set of testing queries, choosing an appropriate training data is extremely important. This is because topics analyzed from this training data directly influence the learning and suggesting performance of our approach. Since the goal of this chapter is to demonstrate that using LDA can improve the suggestion quality over previous methods, we only concentrate on finding a practical training set for the testing 800 queries.

Our training data is collected as follows. Given each query, the top 100 search results were offered by Google API [2]. The contents of these search results are downloaded from the Web. After removing HTML tags and stop words, we stem the left terms and use the traditional term frequency (TF) as term weighting scheme. Finally, we get 66,684 Web pages and 686,295 distinct terms which are used as our training data. We suppose that the crawled search results of those 800 queries working as a small-scale Web archive can cover the content topics that are relevant to the queries themselves.

In experiments, we also empirically studied the effect of the different number of top search results varying from 20 to 100 with step 20. Using top 100 search results as

[1] http://www.acm.org/sigs/sigkdd/kddcup/index.php
[2] http://code.google.com/apis/soapsearch/

training data shows the best performance. Although we can use the public universal Web sources such as Wikipedia, ODP, and so on as training datasets, our experiences on LDA say that training the topic model for such a large dataset is quite time-consuming, let alone we have to estimate again and again with different input parameter values to find a desired model. Moreover, our experimental results show that our training data works well as we expect.

Our evaluation metric is $Precision@N$ which computes the percentage of related queries at the top N queries of a suggestion list. Query suggestion actually is a ranking problem where MAP and $NDCG$ [62] are popular evaluation metrics, especially for the evaluation of Web search. Because the length of a keyword-based query is much shorter than that of a Web page, Web users can read the top 10 or 20 recommended queries much more quickly than they browsing and clicking the top Web pages one by one. In the evaluation of query suggestion, we are more interested in the number of related queries in a suggestion list, i.e., $Precision$ than the order in which the queries are returned such as MAP, $NDCG$ and so on. In KDD cup 2005 data, each query is labeled at most five topic categories by three human labellers, if two queries share at least a same category, we regard them as related queries. In addition, when one query is used as an input query, the left 799 queries are as the past candidate queries, and our approach outputs a suggestion list. We report the experimental results averaged by 800 queries and the three labellers in the following.

7.5.2 Results and Discussions

We compare our approach with two baselines, the query term based suggestion and the pseudo relevance based suggestion. We do not consider the implicit relevance based suggestion which requires click information beforehand. The reasons are out of two aspects: one is that the click data are not available when the query is not submitted before, and the other is the click information is kept by search engines and sometime not easily accessible by public at the current stage.

7.5.2.1 Comparison with Query Term based Suggestion

We conduct experiments by varying the number of latent topics (K) and get the evaluation results of *precision* at top N=5, 10, and 15. The parameter study on K says that our approach is able to achieves the overall highest suggestion precision at $K = 75$. First, we show the suggestion performance of our approach by fixing K=75 at different values of N in Figure 7.3 . The naïve method as our baseline is the query term based suggestion strategy discussed in Section 7.1, which uses the popular Jaccard similarity

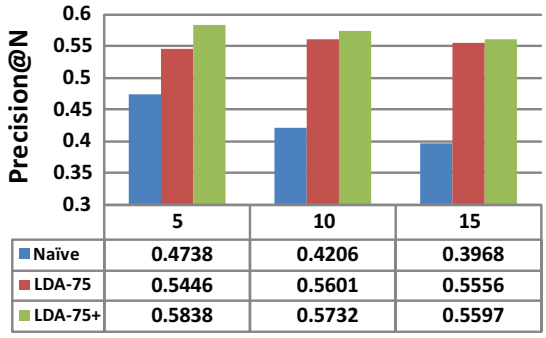

FIGURE 7.3: Precision@N (N=5, 10,15)

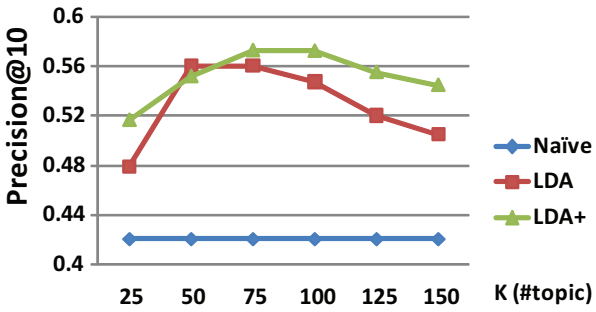

FIGURE 7.4: Precision@10 varying the number of topics

measure based on the term overlap of two queries. For example, the similarity between "car sales" and "automobile sales" is 1/3=0.33. "LDA-75" represents the expression of a query using the inferred hidden topics alone at K=75 and "LDA-75+" represents the combined expression of an input query by integrating the topic assignment with the query term presence at K=75.

From Figure 7.3, we see that our approach significantly improves the suggestion precision over the naïve method. At different values of N, the improvement percentages of "LDA-75" are 14.9%, 33.2%, and 40%, respectively while the improvement percentages of "LDA-75+" are 23.2%, 36.3%, and 41.1%, respectively. In addition, the improvement of our approach becomes larger with the increasing of the value of N, which verifies the intrinsic inability of the naïve method because it simply thinks that Web queries that share common terms are related by lexical occurrence. The hidden topics inferred by LDA help us understand Web queries at a semantic level, thus producing better suggestion results than the baseline. When integrating the hidden topics with the query term vector representation, the suggestion results are further improved. We can say that both semantic topics and lexical occurrence contribute to the improved suggestion performance.

We also show the effect of the number selection of the inferred topics (K) on suggestion precision in Figure 7.4. Due to the page limit, we only report the results of $Precision$@10

TABLE 7.2: Precision@10 of pseudo relevance based suggestion

#top snippets	20	40	60	80	100
cosine	0.5701	0.587	**0.5889**	0.5824	0.5657

which are similar to those of *Precision*@5 and *Precision*@15. As shown in Figure 7.4, the curve of precision scores is not monotonously increasing or decreasing when the number of topics (K) becomes larger. When K is reaching 75, our approach generally achieves the highest precision on our testing query data. Our results say that the larger value of K is not the better. Also, the selection of the parameter K heavily depends on various applications.

7.5.2.2 Comparison with Pseudo Relevance based Suggestion

Likewise we can get two kinds of features to express queries from the search results of queries, i.e, content-based and URL-based. Then, the relatedness between queries can be estimated using these features. We report our study on this issue below.

Using the URL-based features, in Chapter 6, we proposed a query-URL bipartite based approach to improve the performance of traditional pseudo relevance based methods and evaluated it on the same testing data set as the one used here, i.e., KDD cup 2005 data. The highest precision score at top 10 they gained is 0.4675. From Figure 7.4, our result is 0.5732 produced by "LDA-75+", achieving 22.6% improvement.

Given the 800 testing queries, we downloaded the snippets and titles of their top search results as the content-based features to represent them and the computation of query similarity uses the cosine similarity measure by the traditional TF*IDF term weighting scheme, which is called "snippet-cosine" suggestion. Its suggestion results are given in Table 7.2 in terms of *precision*@10. The highest precision score of "snippet-cosine" is 0.5889 when using the snippets of the top 60 search results while our best results is 0.5732. Notice that "snippet-cosine" has to download the snippets of search results in the online suggestion step. Our approach represents Web queries in a low-dimensional latent topic space instead of a high-dimensional snippet term space and does not require any external sources in the online step, but it still shows comparative performance with "snippet-cosine".

The reported experiment results tell us that using Web as a knowledge base is an effective way to learn the meaning of Web queries, thus greatly improving the precision of query suggestion. Although the training step of our approach costs much time, it is completed in an offline processing. We can then apply the well-trained model to various queries as

long as the training set is rich enough to cover the relevant topics of queries, which is an outstanding advantage that is very useful for developing practical realtime applications.

7.6 Summary

In this chapter, we presented a novel approach to related query suggestion using hidden topic model. Unlike previous methods, our approach not only understands Web queries by using Web as a rich knowledge base, but also provides the hidden topics of Web queries to represent their semantic meanings. Moreover, based on the learned topic model using LDA we can give accurate suggestion for a short query without requiring external sources no matter whether the input query was appeared in the past query archive or not. The experimental results on a benchmark dataset containing 800 categorized queries clearly demonstrated that our approach significantly outperforms two baselines in terms of suggestion precision.

Chapter 8

Conclusions and Future Work

The theme of this thesis is to study two strategies which can enhance Web search, a popular information access service on the Web. We have proposed a series of novel approaches. Experimental results validate the effectiveness of our approaches. In this chapter, we summarize our work and point out some potential research directions.

8.1 Conclusions

Many advanced Web searching techniques have been developed and used in commercial Web search engines to find relevant information given a keyword based query. In spite of the improvement of searching accuracy with the development of technologies, it is not always true that Web users can hit upon the proper search queries. In this thesis, we discussed personalized re-ranking of search results by automatically learning user preferences, and suggesting related keywords to help users improve their search experiences.

- In Chapter 3, we designed independent models for short-term and long-term user preferences to consist of a user profile, devised dynamic strategies to the accumulation and degradation changes of user preferences including the content and the structure, and proposed a query and time independent re-ranking mechanism. We empirically studied two kinds of combination methods to re-rank search results because user preferences usually include several attributes. One is score combination and the other is rank aggregation. We observed that some rank-based aggregation methods performed better than the *Socre-Based* method and the *Footrule_S* method performed best in our evaluation.

- In Chapter 4 we presented a query-centric STAR framework for selective utilization of user search behaviors and re-ranking. We showed how our STAR framework can carefully choose and select from the relevant click records the most useful user context for a given input query, and how we applied hierarchical semantic similarity measures in our re-rank strategies. We found that using ODP metadata can effectively learn user-specific query-dependent personalization preference and significantly improve the accuracy of personalized search.

On the other hand, suggesting related keyword based queries can also help Web users get their desired Web pages through search query reformation. We studied three classes of suggestion strategies and conducted extensive experiments to compare the suggestion quality of various feature spaces.

- In Chapter 5 we proposed a novel Web community feature space to enrich search queries. First, the clicked search results of search queris are extracted from Japanese Web access logs. Then, we found the Web community IDs of these clicked search results based on our previous work. The relatedness between queries based on the common Web communities they share produces much higher precision than the traditional URL feature space. It is worth to note that although using the content of these clicked search results can find more related search queries than the URL and Web community spaces, it is not generally available in the case of non-text pages. Therefore, the noun feature space can be used as a supplementary to further enhance suggestion performance.

- In Chapter 6 we presented QBUiC, a query-URL bipartite graph based approach to query recommendation. Different from Chapter 5, we enrich search queries by their top search results instead of their clicked search results. And inspired by Chapter 5, we are still interested in improving the suggestion precision of the traditional URL feature space. We built a query-URL bipartite graph by retrieving search results of queries from a search, generated the query affinity graph using the query-URL vector based similarity measures, and extracted connected components from the query affinity graph. In addition, we devised a novel rank mechanism for ordering the related queries based on the merging distances of a hierarchical agglomerative clustering (HAC). Our rank mechanism is able to capture the propagation of similarity from query to query by inducing an implicit topical relatedness between queries.

- In Chapter 7 we proposed a novel two-step approach to query suggestion. In the offline model-learning step, we collect an appropriate training data by gaining external knowledge from Web to build a generative probabilistic topic model for Web

queries. The training set is carefully selected by considering the topic consistency with queries. Then, for each past query, its posterior distribution of hidden topics is inferred from the learned model. In the online query suggestion step, the same inference is done for an input query. We estimate the similarity between the input query and past queries based on their corresponding topic distributions. Finally, a suggestion list of past queries is ordered according to the similarity scores, which gives the hint to search engine users in rephrasing their query formulation. The probabilistic topic model utilized by our approach makes the sparse query words more related as well as expand the coverage of suggestion to handle future queries better.

8.2 Future Work

For personalized search, in the future we plan to put our methods into larger datasets, and further mine more user-centered information and optimize Web searches in terms of user satisfaction. We can also study more user implicit information to construct the user profile, such as the time interval between two clicks, browsing patterns, and so on. Our ongoing research also includes designing an effective updating policy for user profiles, and more effective rank aggregation methods for further optimization of personalized search.

For related keyword suggestion, our research continues along several dimensions. We are designing an optimization algorithm for incremental updates of related query data and evaluating other similarity measures. we plan to study the effect of different Web corpora on query suggestion. Moreover, we can easily adjust our approach to the Question & Answer system which matched the question issued by a user to the most relevant question previously answered by human experts. We will evaluate the effect of related queries on the quality of Web information retrieval since the goal of finding related queries tries to help users get their desired information by formulating better search queries.

List of Publication

International Conference Papers

1. Lin Li, Zhenglu Yang, Botao Wang, and Masaru Kitsuregawa. Dynamic adaptation strategies for long-term and short-term user profile to personalize search. In *Proceedings of A Joint Conference of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management, (APWeb/WAIM'07)*, pages 228–240, Huang Shan, China, 2007.

2. Lin Li, Zhenglu Yang, and Masaru Kitsuregawa. Aggregating user-centered rankings to improve web search. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence, (AAAI'07)*, pages 1884–1885, Vancouver, British Columbia, Canada, 2007 (poster).

3. Lin Li, Zhenglu Yang, Kulwadee Somboonviwat, and Masaru Kitsuregawa. User-assisted similarity estimation for searching related web pages. In *Proceedings of the 18th ACM Conference on Hypertext and Hypermedia, (HT'07)*, pages 11–20, Manchester, UK, 2007

4. Lin Li, Zhenglu Yang, and Masaru Kitsuregawa. Using ontology-based user preferences to aggregate rank lists in web search. In *Proceedings of Advances in Knowledge Discovery and Data Mining, 12nd Pacific-Asia Conference, (PAKDD'08)*, pages 923–931, Osaka, Japan, 2008 (short paper).

5. Lin Li, Zhenglu Yang, L. Liu, and Masaru Kitsuregawa. Query-url bipartite based approach to personalized query recommendation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, (AAAI'08)*, pages 1189–1194, Chicago, Illinois, USA, 2008.

6. Lin Li, Shingo Otsuka, and Masaru Kitsuregawa. Query recommendation using large-scale web access logs and web page archive. In *Proceedings of 19th International Conference on Database and Expert Systems Applications, (DEXA'08)*, pages 134–141, Turin, Italy, 2008 (short paper).

Domestic Workshop Papers and Reports

1. Lin Li and Masaru Kitsuregawa. Personalizing web search via modelling adaptive user profile. *National Data Engineering WorkShop, (DEWS'07)*, pages M5–5, Hiroshima, Japan, 2007.

2. Lin Li, Kulwadee Somboonviwat, and Masaru Kitsuregawa. Aggregating user-centered rankings to improve web search. *Forum on Web and Database, (DB-Web'07)*, pages 1A–2, Tokyo, Japan, 2007.

3. Lin Li and Masaru Kitsuregawa. Finding latent neighbors for query recommendation: a user-Controllable scheme. *National Data Engineering WorkShop, (DEWS'08)*, pages B3–5, Miyazaki, Japan, 2008.

4. Lin Li, Shingo Otsuka, and Masaru Kitsuregawa. Utilizing related web queries to enhance web search. *Forum on Web and Database, (DBWeb'08)*, Tokyo, Japan, 2008 (poster).

5. Lin Li, Zhenglu Yang, and Masaru Kitsuregawa. Rank optimization of personalized search. *Forum on Data Engineering and Information Management, (DEIM'09)*, pages A9-5, Shizuoka, Japan, 2009.

Bibliography

[1] J. A. Aslam and M. H. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,(SIGIR'01)*, pages 275–284, New Orleans, Louisiana, USA, 2001.

[2] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. Improving search engines by query clustering. *JASIST*, 58(12):1793–1804, 2007.

[3] E. Balfe and B. Smyth. An analysis of query similarity in collaborative web search. In *Advances in Information Retrieval, 27th European Conference on IR Research, (ECIR'05)*, pages 330–344, Santiago de Compostela, Spain, 2005.

[4] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*, pages 131–140, Banff, Alberta, Canada, 2007.

[5] D. Beeferman and A. L. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD'00)*, pages 407–416, Boston, MA, USA, 2000.

[6] D. Billsus and M. J. Pazzani. A hybrid user model for news story classification. In *Proceedings of the 7th International Conference on User modeling (UM'99)*, pages 99–108, Secaucus, NJ, USA, 1999.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[8] D. Boley. Principal direction divisive partitioning. *Data Min. Knowl. Discov.*, 2(4):325–344, 1998.

[9] J. Borda. Mémoire sur les élections au scrutin. *Comptes rendus de l'Académie des sciences*, 44:42–51, 1781.

[10] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart. In *Proceedings of Text REtrieval Conference (TREC'03)*, pages 69–080, Gaithersburg, Maryland, 1994.

[11] P. Calado, M. Cristo, M. A. Gonçalves, E. S. de Moura, B. A. Ribeiro-Neto, and N. Ziviani. Link-based similarity measures for the classification of web documents. *JASIST*, 57(2):208–221, 2006.

[12] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (KDD'08)*, pages 875–883, Las Vegas, Nevada, USA, 2008.

[13] L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world-wide web. *Computer Networks and ISDN Systems*, (27(6)), 1995.

[14] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th international conference on World Wide Web, (WWW'05)*, pages 2–11, Chiba, Japan, 2005.

[15] P.-A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, pages 7–14, Amsterdam, The Netherlands, 2007.

[16] P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using ODP metadata to personalize search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, pages 178–185, Salvador, Brazil, 2005.

[17] K. Collins-Thompson and J. Callan. Query expansion using random walk models. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management (CIKM'05)*, pages 704–711, Bremen, Germany, 2005.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. McGraw-Hill, 1990.

[19] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Query expansion by mining user logs. *IEEE Trans. Knowl. Data Eng.*, 15(4):829–839, 2003.

[20] M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *J. Algorithms*, 41(2):388–403, 2001.

[21] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *Computer Networks*, 31(11-16):1467–1479, 1999.

[22] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 89–98, Washington, DC, USA, 2003.

[23] P. Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(2):262–268, 1977.

[24] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*, pages 581–590, Banff, Alberta, Canada, 2007.

[25] S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: A system for personal information retrieval and re-use. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*, pages 72–79, Toronto, Canada, 2003.

[26] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*, pages 613–622, Hong Kong, China, 2001.

[27] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03)*, pages 301–312, San Diego, California, USA, 2003.

[28] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Proceedings of the 14th International Conference on World Wide Web - Special interest tracks and posters (WWW'06)*, pages 801–810, Chiba, Japan, 2005.

[29] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: Social searching? In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 306–313, Philadelphia, PA, USA, 1997.

[30] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Proceedings of the 6h ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD'00)*, pages 150–160, Boston, MA, USA, 2000.

[31] B. M. Fonseca, P. B. Golgher, B. Pôssas, B. A. Ribeiro-Neto, and N. Ziviani. Concept-based interactive query expansion. In *Proceedings of the 2005 ACM*

CIKM International Conference on Information and Knowledge Management, (CIKM'05), pages 696–703, 2005.

[32] D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HT'98)*, pages 225–234, Pittsburgh, PA, USA, 1998.

[33] N. S. Glance. Community search assistant. In *Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI'01)*, pages 91–96, Santa Fe, NM, USA, 2001.

[34] Google Directory. http://directory.google.com.

[35] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[36] M. Hansen and E. Shriver. Using navigation data to improve ir functions in the context of web search. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management (CIKM'01)*, pages 135–142, Atlanta, Georgia, USA, 2001.

[37] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

[38] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 289–296, Stockholm, Sweden, 1999.

[39] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.

[40] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, pages 41–48, Athens, Greece, 2000.

[41] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 538–543, Edmonton, Alberta, Canada, 2002.

[42] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*, pages 271–279, Budapest, Hungary, 2003.

[43] H. R. Kim and P. K. Chan. Learning implicit user interest hierarchy for context in personalization. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces (IUI'03)*, pages 101–108, Miami, FL, USA, 2003.

[44] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[45] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar. Searching with context. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*, pages 477–486, Edinburgh, Scotland, UK, 2006.

[46] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31(11-16):1481–1493, 1999.

[47] Y. Labrou and T. W. Finin. Yahoo! As an ontology: Using Yahoo! categories to describe documents. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management (CIKM'99)*, pages 180–187, Kansas City, Missouri, USA, 1999.

[48] W. Lam, S. Mukhopadhyay, J. Mostafa, and M. J. Palakal. Detection of shifts in user interests for personalized information filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 317 – 325, Zurich, Switzerland, 1996.

[49] G. N. Lance and W. T. Williams. A generalized sorting strategy for computer classifications. *Nature*, 212:218, 1966.

[50] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies: 1. hierarchical systems. *The Computer Journal*, 9:373–380, 1967.

[51] G. Lebanon and J. D. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *Proceedings of the 19th International Conference on Machine Learning, (ICML'02)*, pages 363–370, Sydney, Australia, 2002.

[52] L. Li, S. Otsuka, and M. Kitsuregawa. Query recommendation using large-scale web access logs and web page archive. In *Proceedings of 19th International Conference on Database and Expert Systems Applications (DEXA'08)*, pages 134–141, Turin, Italy, 2008.

[53] L. Li, Z. Yang, and M. Kitsuregawa. Aggregating user-centered rankings to improve web search. In *Proceedings of the 22nd AAAI Conf. on Artificial Intelligence (AAAI'07)*, pages 1884–1885, Vancouver, British Columbia, Canada, 2007.

[54] L. Li, Z. Yang, and M. Kitsuregawa. Using ontology-based user preferences to aggregate rank lists in web search. In *Proceedings of Advances in Knowledge Discovery and Data Mining, 12nd Pacific-Asia Conference (PAKDD'08)*, pages 923–931, Osaka, Japan, 2008.

[55] L. Li, Z. Yang, L. Liu, and M. Kitsuregawa. Query-url bipartite based approach to personalized query recommendation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence,(AAAI'08)*, pages 1189–1194, Chicago, Illinois, USA, 2008.

[56] L. Li, Z. Yang, B. Wang, and M. Kitsuregawa. Dynamic adaptation strategies for long-term and short-term user profile to personalize search. In *Proceedings of A Joint Conference of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management*, pages 228–240, Huang Shan, China, 2007.

[57] Y. Li, Z. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. Knowl. Data Eng.*, 15(4):871–882, 2003.

[58] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–, 1991.

[59] F. Liu, C. T. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Trans. Knowl. Data Eng.*, 16(1):28–40, 2004.

[60] Y. Lv, L. Sun, J. Zhang, J.-Y. Nie, W. Chen, and W. Zhang. An iterative implicit feedback approach to personalized search. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*, pages 585–592, Sydney, Australia, 2006.

[61] H. Ma, H. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, (CIKM'08)*, pages 709–718, Napa Valley, California,USA, 2008.

[62] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[63] Q. Mei, D. Zhou, and K. W. Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, (CIKM'08)*, pages 469–478, Napa Valley, California, USA, 2008.

[64] M. H. Montague and J. A. Aslam. Relevance score normalization for metasearch. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management (CIKM'01)*, pages 427–433, Atlanta, Georgia, USA, 2001.

[65] M. H. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management (CIKM'02)*, pages 538–548, McLean, VA, USA, 2002.

[66] N. Nanas, V. S. Uren, and A. N. D. Roeck. Building and applying a concept hierarchy representation of a user profile. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*, pages 198–204, Toronto, Canada, 2003.

[67] S. Otsuka and M. Kitsuregawa. Clustering of search engine keywords using access logs. In *Proceedings of 17th International Conference on Database and Expert Systems Applications (DEXA'06)*, pages 842–852, Kraków, Poland, 2006.

[68] S. Otsuka, M. Toyoda, J. Hirai, and M. Kitsuregawa. Extracting user behavior by web communities technology on global web logs. In *Proceedings of 15th International Conference on Database and Expert Systems Applications (DEXA'04)*, pages 957–968, Zaragoza, Spain, 2004.

[69] S. Otsuka, M. Toyoda, and M. Kitsuregawa. A study for related words finding methods using global web access logs. *IPSJ Transactions on Databases (TOD)*, 46:82–92, 2005.

[70] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[71] F. C. N. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 183–190, 1993.

[72] A. Pretschner and S. Gauch. Ontology based personalized search. In *Proceedings of the 9th International Conference on Tools with Artificial Intelligence (ICTAI'99)*, pages 391–398, Chicago, Illinois, USA, 1999.

[73] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*, pages 727–736, Edinburgh, Scotland, 2006.

[74] H. rae Kim and P. K. Chan. Personalized ranking of search results with learned user interest hierarchies from bookmarks. In *Proceedings of the 7th WEBKDD workshop on Knowledge Discovery from the Web (WEBKDD'05)*, pages 32–43, Chicago, Illinois, USA, 2005.

[75] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 344–350, Seattle, Washington, USA, 1995.

[76] M. Rege, M. Dong, and F. Fotouhi. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM '06)*, pages 532–541, Hong Kong, China, 2006.

[77] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC'03)*, pages 841–846, Melbourne, FL, USA, 2003.

[78] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic (NIPS'01)*, pages 1441–1448, Vancouver, British Columbia, Canada, 2001.

[79] D. E. Rose and D. Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, pages 13–19, New York, NY, USA, 2004.

[80] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297, 1990.

[81] V. Schickel-Zuber and B. Faltings. Inferring user's preferences using ontologies. In *Proceedings of The 21st National Conf. on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference (AAAI'06)*, pages 1413–1418, Boston, Massachusetts, USA, 2006.

[82] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, pages 43–50, Salvador, Brazil, 2005.

[83] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *Proceedings of the 2005 ACM CIKM Int'l Conf. on Information and Knowledge Management (CIKM'05)*, pages 824–831, Bremen, Germany, 2005.

[84] X. Shi and C. C. Yang. Mining related queries from web search engine query logs using an improved association rule mining model. *JASIST*, 58(12):1871–1883, 2007.

[85] M. Shokouhi. Segmentation of search engine results for effective data-fusion. In *Proceedings of the 29th European Conference on IR Research (ECIR'07)*, pages 185–197, Rome, Italy, 2007.

[86] S. Siegel and N. J. Castellan. *Nonparametric Statistics for the Behavioral Sciences.* McGraw-Hill Book Co., second edition, 1988.

[87] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, pages 208–215, Athens, Greece, 2000.

[88] S. J. Soltysiak and I. B. Crabtree. Automatic learning of user profiles- towards the personalisation of agent services. *BT Technology Journal*, 16(3):110–117, 1998.

[89] M. Speretta and S. Gauch. Personalized search based on user search histories. In *Proceedings of the IEEE / WIC / ACM International Conference on Web Intelligence (WI'05)*, pages 622–628, Compiegne, France, 2005.

[90] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, pages 418–425, Houston, Texas, USA, 2005.

[91] R. Sun, C.-H. Ong, and T.-S. Chua. Mining dependency relations for query expansion in passage retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*, pages 382–389, Seattle, Washington, USA, 2006.

[92] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 718–723, Philadelphia, PA, USA, 2006.

[93] F. Tanudjaja and L. Mu. Persona: A contextualized and personalized web search. In *Proceedings of the 35th Hawaii Int'l Conf. on System Sciences (HICSS'02)*, pages 67–75, 2002.

[94] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th Annual Int'l ACM*

SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'05), pages 449–456, Salvador, Brazil, 2005.

[95] M. Toyoda and M. Kitsuregawa. Creating a web community chart for navigating related communities. In *Proceedings ofthe 12th ACM Conference on Hypertext and Hypermedia (HT'01)*, pages 103–112, Århus, Denmark, 2001.

[96] E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 61–69, Dublin, Ireland, 1994.

[97] J.-R. Wen, J.-Y. Nie, and H. Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1):59–81, 2002.

[98] D. H. Widyantoro, T. R. Ioerger, and J. Yen. Learning user interest dynamics with a three-descriptor representation. *JASIST*, 52(3):212–225, 2001.

[99] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 4–11, Zurich, Switzerland, 1996.

[100] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, 18(1):79–112, 2000.

[101] J.-M. Yang, R. Cai, F. Jing, S. Wang, L. Zhang, and W.-Y. Ma. Search-based query suggestion. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, (CIKM'08)*, pages 1439–1440, Napa Valley, California,USA, 2008.

[102] Z. Yang, L. Li, and M. Kitsuregawa. Efficient querying relaxed dominant relationship between product items based on rank aggregation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 1261–1266, Chicago, Illinois, USA, 2008.

[103] H. P. Young. Condorcet's theory of voting. *American Political Science Review*, 82(4):1231–1244, 1988.

[104] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Bipartite graph partitioning and data clustering. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'01)*, pages 25–32, 2001.

[105] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on World Wide Web (WWW'06)*, pages 1039–1040, Edinburgh, Scotland, UK, 2006.

[106] S. Zhu, Q. Fang, X. Deng, and W. Zheng. Metasearch via voting. In *Proceedings of the 4th Int'l Conf. on Intelligent Data Engineering and Automated Learning (IDEAL'03)*, pages 734–741, Hong Kong, China, 2003.

[107] Y. Zhu and L. Gruenwald. Query expansion using web access log files. In *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA'05)*, pages 686–695, Copenhagen, Denmark, 2005.

[108] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998.