
Query Clustering in the Web Context

Ji-Rong Wen and Hong-Jiang Zhang
Microsoft Research Asia
3F, Beijing Sigma Center
No. 49, Zhichun Road, Haidian District
Beijing, P.R. China 100080
E-mail: {jrwen, hjzhang}@microsoft.com

Contents

1	Introduction	2
2	Content-based Query Clustering	5
2.1	Similarity Based on Keywords or Phrases	5
2.2	Similarity Based on Word-Level String Matching	6
2.3	Similarity Based on Character-Level String Matching	7
3	Session-based Query Clustering	8
3.1	Query Session Extraction	10
3.2	Naive Similarity Based on Clickthroughs	12
3.3	Combination of Query Clustering and Document Clustering	13
3.3.1	Unidirectional Clustering	14
3.3.2	Bidirectional Clustering	15
3.4	Clustering Queries through Query Co-occurrences	17
4	Combining Evidences from Query Content and Query Session	18
5	Choosing a Clustering Algorithm	19
6	A Case Study on Encarta Online Search Engine	21
6.1	Query Sessions	22
6.2	Document Collection and Document Similarity	22
6.3	Evaluation of Different Similarity Functions	23
7	Summary	26

References

1 Introduction

Query clustering is a class of techniques aiming at grouping users' semantically related, not syntactically related, queries in a query repository, which were accumulated with the interactions between users and the system. While there are numerous previous works on document clustering, query clustering is a relatively new topic. The driving force of the development of query clustering techniques comes recently from the requirements of modern web searching. Below we briefly analyze several motivations and applications of query clustering — FAQ detecting, index-term selection and query reformulation.

FAQ Detecting Faced with the increasing requirement for more precise information retrieval systems, a new generation of search engines — or question answering systems — has emerged on the Web (e.g. AskJeeves, <http://www.askjeeves.com>). Unlike traditional search engines, these new systems try to "understand" user's questions in order to suggest similar questions that other people often ask and for which the system has the correct answers. In fact, the correct answers have been prepared or checked by human editors in most cases. It is then guaranteed that, if one of the suggested questions is truly similar to that of the users, the answers provided by the system will be relevant. The assumption behind such a system is that many people are interested in the same questions — the Frequently Asked Questions (FAQs). It is assumed that if the system can correctly understand or identify these questions, then a significant majority of users' questions will be answered precisely. The human editors have to detect which questions/queries are FAQs from users' query logs so that they can prepare/check their answers. However, this is a tedious task. What is needed is an automatic tool that helps the editors to identify FAQs, so that the editors only need to decide the answers to those FAQs. This is exactly a query clustering problem — to cluster similar queries/questions together in order to discover FAQs. However, the queries submitted by web users are very different, and they are not always well-formed questions. Queries with the same or similar search intentions may be represented with diverse words. Therefore, some measures are needed to estimate the similarity between queries, which is the core problem of query clustering.

Index-Term Selection Despite the fact that keywords are not always good descriptors of the contents, most existing search engines still solely rely on keywords contained in web pages to construct indices. This is one of the key factors that affect the precision of search engines. Inconsistency between term usages in queries and documents is a well-known problem in information retrieval, which was first observed by Furnas [9]. It is even worse when the query is very short as often the case on the web. Through web logs, [3] testified the "word mismatching" assumption in quantity. The test results show that there is indeed a large difference between the query terms and document terms, and the traditional way of directly extracting index terms from documents will not be effective when the user submits queries containing terms different from those in the documents. Query clustering is a promising technique to provide a solution to the word mismatching problem. If similar queries can be recognized and clustered together, the resulting query clusters will be very good sources for selecting additional index terms for documents. For example, if queries "atomic bomb", "Manhattan Project" "Hiroshima bomb" and "nuclear weapon" are put into a query cluster, this cluster, not the individual terms, can be used as a whole to index documents related to atomic bomb. In this way, any queries contained in the cluster can be linked to these documents.

Query Reformulation The web is not a well-organized information source where innumerable "authors" create their websites independently. Those authors "vocabularies" vary greatly. Moreover, most words in the natural language have inherent ambiguity. These reasons make it rather difficult for the web users to formulate queries with appropriate words. Many web search engines, such as AltaVista (<http://www.altavista.com>), Excite (www.excite.com), Lycos (www.lycos.com), etc., attempt to identify some of the users' intentions and suggest a list of alternate terms for the user to reformulate his/her query. This kind of term suggestion mechanism needs to track the users emerging topics of interest, such as events in the news or newly publicized web sites. Obviously, query clustering, which intends to find related queries from users' daily logs, could be one of the critical techniques to drive this term suggestion process.

For any clustering problem, researches have been concerned mainly with two aspects — similarity functions and algorithms for the clustering process. The key problem underlying query clustering is to determine an adequate similarity function so that truly similar queries can be grouped together

using a clustering algorithm. Currently, there are mainly two categories of methods to calculate the similarity between queries: one is based on query content, and the other on query session.

The classic approach to information retrieval (IR) would suggest a similarity calculation between queries according to their contents (e.g., keywords) [26]. The content-based query clustering methods just follow the basic term-semantic hypothesis in IR: *If two queries contain the same or similar terms, they denote the same or similar information needs.* However, this hypothesis has some known drawbacks due to the limitations of keywords, just as in the case of keyword-based IR. In the case of queries, in particular, the keyword-based similarity calculation will be very inaccurate (with respect to semantic similarities) due to the short length of queries.

To overcome the short-query problem, researchers are looking for other evidences to judge the similarity between queries [28][29][1][12]. The most promising way is to extend the "query" concept to the "query session" concept. A query session is defined as the query and the subsequent activities the user performed. Usually, query sessions can be extracted from the logs of search engines. The basic assumption here is that the activities following a query are relevant to the query and represent, to some extent, the semantic features of the query. The query text and the activities in a query session as a whole can represent the search intention of the user more precisely. Moreover, the ambiguity of some query terms is eliminated in query sessions. For instance, if the user visited a few tour websites after submitted a query "Java", it is reasonable to deduce that the user was searching for information about "Java Island" not "Java programming language". Accordingly, "query clustering" can be extended to "query session clustering". The hypothesis underlying the session-based query clustering methods is formally stated as: *Two queries leading to the similar user activities are similar.* User activities are comparable to user relevance feedback in a traditional IR environment, except that user activities denote implicit and not always valid relevance judgments.

Similarities based on query contents and query sessions represent two different points of view. The two criteria have their own advantages and shortcomings. In general, content-based criterion tend to cluster queries with the same or similar terms; but, due to the ambiguity of words, similar terms could be used to represent different query requirements and different terms may represent the same requirements. Session-based criterion tend to cluster queries related to the same or similar topics; but a session is usually noisy and contains more than one topic or "interest point", thus queries with different intentions may lead to similar activities. Since information

needs may be partially captured by both query contents and user activities, some combined measures taking advantage of both criteria may be more appealing.

2 Content-based Query Clustering

Although the discussion of query clustering is relatively new, there have been extensive studies on document clustering. The first group of related clustering approaches is certainly those that cluster documents using the contents they contain. Like documents, there are different ways to represent query content: keywords, words in their order, and phrases. They provide different measures of similarity.

2.1 Similarity Based on Keywords or Phrases

The first measure of query similarity comes directly from IR studies. In traditional approaches, in general, a document is represented as a vector in a feature space formed by all the keywords [24]. Typically, people use either Cosine-similarity, Jaccard-similarity or Dice-similarity [24] as similarity functions. Keywords are all the words, except the function words included in a stop-list. All the keywords can be stemmed using the Porter algorithm [21]. The keyword-based similarity function is defined as follows:

$$similarity_{keyword}(q_1, q_2) = \frac{KN(q_1, q_2)}{\max(kn(q_1), kn(q_2))} \quad (1)$$

where $kn(\cdot)$ is the number of the keywords in a query, $KN(q_1, q_2)$ is the number of common keywords in two queries.

If query terms are weighted, the Cosine similarity [24] can be used instead:

$$similarity_{weighted-keyword}(q_1, q_2) = \frac{\sum_{i=1}^k cw_i(q_1) \times cw_i(q_2)}{\sqrt{\sum_{i=1}^m w_i^2(q_1)} \times \sqrt{\sum_{i=1}^n w_i^2(q_2)}} \quad (2)$$

where $cw_i(q_1)$ and $cw_i(q_2)$ are the weights of the i -th common keyword in the query q_1 and q_2 respectively, and $w_i(q_1)$ and $w_i(q_2)$ are the weights of the i -th keywords in the query q_1 and q_2 respectively. Usually, $tf * idf$ [24] is used for keyword weighting.

The above measures can be easily extended to phrases. Since phrases are a more precise representation of meaning than single words, if we can identify phrases in queries, we can obtain a more accurate calculation of query

similarity. For example, the two queries "history of China" and "history of the United States" are very close queries (both asking about the history of a country). Their similarity is 0.33 on the basis of keywords. If we can recognize "the United States" as a phrase and take it as a single term, the similarity between these two queries is increased to 0.5. The calculation of phrase-based similarity is similar to Formula (1) and (2). We only need to recognize phrases in queries. There are two main methods for doing this. One is by using a noun phrase recognizer based on syntactic rules and statistics [4][15]. Another way is to use a phrase dictionary. A phrase dictionary can be constructed manually or using an automatic phrase recognizer based on a syntactic and statistical analysis on corpus.

The keyword-based measure has provided interesting results in the document clustering domain. One contributing factor is the large number of keywords contained in documents. Even if some of the keywords of two similar documents are different, there are still many other keywords that can make the documents similar in the similarity calculation. However, since queries, especially the queries submitted to the search engines, are typically very short, in many cases it is hard to deduce the semantics from the queries themselves. Therefore, keywords alone do not provide a reliable basis for clustering queries effectively.

2.2 Similarity Based on Word-Level String Matching

Keyword-based measure treats words such as "where", "who" and "why" as stop words. For questions, however, these words (if they occur) encode important information about the user's need, particularly in the question-answering search engines such as AskJeeves. For example, with a "who"-question, the user intends to find information about a person. So, even if a keyword-based approach is used in query clustering, it should be modified from that used in traditional document clustering. Special attention is paid to such words in Question Answering (QA) [14][25], where they are used as prominent indicators of question type. The whole question is represented as a template in accordance with the question type. During question evaluation, the question template may be expanded using a thesaurus (e.g. WordNet [17]) or morphological transformations.

In the web context, well-formed natural language questions represented only a small portion of queries. Most queries are simply short phrases or keywords (e.g. "population of U.S."). The approach used in QA is therefore not completely applicable to the web query clustering problem. However, if words denoting a question type do appear in a complete question, these

words should be taken into account.

Here we define a similarity measure that uses all the words in the queries for similarity estimation, even the stop words. Comparison between queries becomes an inexact string-matching problem, as formulated by [10]. Similarity may be determined by the edit distance, which is a measure based on the number of edit operations (insertion, deletion, or substitution of a word) necessary to unify two strings (queries). The maximum number of the words in the two queries is used to divide the edit distance so that the value can be constrained within the range of $[0, 1]$. The similarity is inversely proportional to the normalized edit distance:

$$similarity_{edit}(q_1, q_2) = 1 - \frac{edit-distance(q_1, q_2)}{max(wn(q_1), wn(q_2))} \quad (3)$$

where $wn(.)$ is the number of the words in a query.

The advantage of this measure is that it takes into account the word order, as well as words that denote query types such as "who" and "what" if they appear in a query. This method is more flexible than those used in QA systems, which rely on special recognition mechanisms for different types of questions. It is therefore more suitable to the query clustering situation. This measure is particularly useful for long and complete questions in natural language. Below are some questions that have been grouped into one cluster:

- Query 1: How far away is the moon?
- Query 2: How far is the closest star?
- Query 3: How far is the nearest star?

This cluster contains questions of the form "How far (away) is the X".

In the similarity calculations described above, we can further incorporate a dictionary of synonyms. If two words/terms are synonyms, their similarity is set at a predetermined value (e.g., 0.8). It is then easy to incorporate this similarity between synonyms into Formula (1), (2) and (3).

2.3 Similarity Based on Character-Level String Matching

The web user often misspells his/her query because the user does not know the spelling, or because of a typing error, or because the user is not completely sure about the spelling. According to [13], the four error types, insertion, deletion, substitution and transposition encompass 80 percent of

all spelling errors. Some typical errors like misspelling "Pokemon" as "Poke-man" occur very frequently when users using search engines. Query clustering can also be used to detect and correct such a kind of spelling errors in user queries.

The word-level string matching method above is incapable of dealing with the spelling error problem. Instead, just like [13], the character-level string matching method is effective to group a query with its misspelling counterparts. In this case, the edit distance in Formula (3) is defined as the number of edit operations (insertion, deletion, or substitution of a character) necessary to unify two queries.

Normally a search engine does not give any answers to queries when the search words do not exist in the indices. Once we can group the queries with spelling errors together, we find a way to collect the most frequent spelling errors in user queries. Then an automatic query correction mechanism could be employed to help the users search the Web more smoothly, such as the *searchspell* website (<http://www.searchspell.com/typo>). Although query clustering by itself cannot solve the misspelling problem thoroughly, it does provide the website editors a powerful tool to assist them to detect and correct spelling errors more easily.

3 Session-based Query Clustering

Due to the short length and word ambiguity of user queries, content-based clustering methods usually cannot group those semantically related queries. Session-based query clustering is a newly emerging strategy, which employs the evidences in query sessions to deduce and detect users' search intentions. Many search engines have accumulated a large amount of web query logs, from which we can find out what the query is, and the web pages the user has selected to browse. Typically, a query session is made up of the query and the subsequent activities the user performed, which can be extracted from query logs. Instead of relying only on the query terms themselves, session-based query clustering methods take advantage of various evidences hidden in query sessions to calculate the similarity between queries.

One important assumption behind these methods is that the activities following a query are relevant to the query. User activities can be viewed as (implicit) user relevance feedback in a traditional IR environment. At first glance, this assumption may seem to be too strong. Users may click the irrelevant web pages, users may make error-clicks, and users may shift their interests after submitting the query. However, in the web environment, the

choice of a particular web page from a list by a user is a good indication of the user's search intention. Although it is not as accurate as explicit relevance judgment in traditional IR, the user's choice does suggest a certain degree of "relevance" of that web page to his information need. Users usually have rough ideas about what they want when submitting queries. So when they get a list of web pages, users do not select them randomly. Instead, they click and read those web pages which are the most similar to what are in their minds based on the information provided by the search engine on the returned candidate answers (e.g. titles and snippets). Therefore, these clicked web pages do have some relationship with the queries they submit. Even if some of the web page clicks are erroneous, we can expect that most users do click on web pages that are relevant.

In fact, using cross-reference between queries and user activities to improve retrieval performance is not a new idea in the IR field. Relevance feedback in IR is a typical exploitation of cross-references [22][23], which modifies a query based on user's relevance judgments of the retrieved documents. Relevance feedback can achieve very good performance if the users provide sufficient and correct relevance judgments. Unfortunately, in a real search context, users usually are reluctant to provide such relevance feedback information. Therefore, relevance feedback is seldom used by the commercial search engines. To overcome the difficulty due to the lack of sufficient relevance judgments, pseudo-relevance feedback (also known as blind feedback) is commonly used. Local feedback mimics relevance feedback by assuming the top-ranked documents to be relevant [31]. Nevertheless, this method has an obvious drawback: if a large fraction of the top-ranked documents is actually irrelevant, the performance of this method is likely to be worse. Now let's look at query logs. Query logs can be taken as a very valuable resource containing abundant relevance feedback data, which is not available in the traditional IR environment. With the query logs, we can overcome the problem of lacking sufficient relevance judgments in traditional relevance feedback technique. In comparison with pseudo-relevance feedback, the query session based method has an obvious advantage: not only are the clicked web pages part of the top-ranked web pages, but also there is a further selection by the user. So clicks are more reliable indications than those used in pseudo relevance feedback.

Similar ideas have been used in some other work in IR. Voorhees et al. [27] try to evaluate the quality of an IR system for a cluster of queries on different document collections. Their goal is to determine an appropriate method for database merging according to the quality estimation. The assumption used in query clustering is that if two queries retrieve many

documents in common they are on the same topic. In [16], a measure of query similarity is used to route queries to proper collection replicas. In their work, two queries are considered to be related to the same topic if the top 20 documents they retrieve completely overlap.

3.1 Query Session Extraction

Typically, web logs will track the activities of the users who are utilizing a search engine. Each record in the web log includes the IP address of the client computer, timestamp, the URL of the requested item, type of web browser, protocol, etc. The web log of a search engine records various kinds of user activities, such as getting HTML pages, getting images and running Java scripts. Although all these activities reflect, more or less, a user's intention, the query terms and the web pages the user visited are the most solid evidences to guess users' search intention. Therefore, a query session is the query a user submitted to a search engine together with the web pages he/she visits in response to his/her request. Table 1 lists a small excerpt of query log from the Encarta web site (We omit some fields such as the type of web browser, the IP address of the server, the bytes sent or received, etc.). This query session shows that a user submitted a query "atomic bomb" to the search engine and then browsed two web pages, which are two Encarta articles related to atomic bomb.

Table 1: Sample Query Session

IP address	Cookie	Timestamp	URL
999.999.999.1	FF4AF1D6935A	4/25/00:2:14:14	http://encarta.msn.com/ find/search.asp?s=atomic+bomb
999.999.999.1	FF4AF1D6935A	4/25/00:2:14:38	http://encarta.msn.com/ article/05808000.html
999.999.999.1	FF4AF1D6935A	4/25/00:2:21:45	http://encarta.msn.com/ article/01FC9000.html

Since the HTTP protocol requires a separate connection for every client-server interaction, the activities of multiple users usually interleaved with each other. There are no clear boundaries among user query sessions in the logs, which makes it a difficult task to extract individual query sessions from web query logs, as addressed in [2][11][20].

Query session extraction mainly contains two steps — user identification and session identification. User identification is the process of isolating from

the logs the activities associated with an individual user. The most intuitive way of user identification is to group the activities with the same IP address. But IP address is not a reliable indication to distinguish users in the case of the broad existence of local caches, corporate firewalls, and proxy servers. In a web server log, all requests from a proxy server have the same identifier, even though the requests are potentially from more than one user. Also, due to proxy server level caching, a single request from the server could actually be viewed by multiple users. [2][19] proposed some heuristics to distinguish different users with the same IP address. For example, if the agent log shows a change in browser software or operating system, a reasonable assumption to make is that each different agent type for an IP address represents a different user. Another heuristic for user identification is to use the access log in conjunction with site topology to construct browsing paths for each user. That is if a requested page is not directly reachable by a hyperlink from any of the pages visited by the user, the heuristic assumes that there is another user with the same IP address. However, these heuristics are not "perfect" solution to the user identification problem. Two users with the same IP address that use the same browser on the same type of machine can easily be confused as a single user if they are looking at the same set of pages. Cookies are markers that are used to tag and track site visitors automatically. Cookies are more reliable than IP addresses because, in general, the cookies from two computers are different, even when they use the same IP address. But cookies can be deleted by the user since he/she doesn't want the server to track his/her behaviors. Another method to identify users is through user ID. User ID has the advantage of being able to collect additional demographic information beyond what is automatically collected in the server log, as well as simplifying the identification of user sessions. However, again due to privacy concerns, many users choose not to browse sites that require registration and logins. There is a constant struggle between the web user's desire for privacy and the web provider's desire for information about who is using their site. Many users choose to disable the browser features that enable these methods.

The goal of session identification is to divide the queries and page accesses of each user into individual sessions. Finding the beginning of a query session is trivial: a query session begins when a user submits a search to a search engine. However, it is more difficult to determine when a search session ends. The simplest method of achieving this is through a timeout, where if the time between page requests exceeds a certain limit, it is assumed that the user is starting a new session. Many commercial products use 30 minutes as a default timeout. Another way is to get the timeout

through analyzing empirical data. Once a site log has been analyzed and usage statistics obtained, a timeout that is appropriate for the specific Web site can be fed back into the session identification algorithm.

3.2 Naive Similarity Based on Clickthroughs

The first session-based similarity function is to simply use clicked pages as the semantic descriptions of the user queries and considers each web page in isolation. This query clustering approach is based on the following principle: *If users clicked on the same web pages for different queries, then these queries are similar.* The formal description of this similarity function is as follows:

Let $D(q_1)$ and $D(q_2)$ be the set of web pages the system presents to the user as search the results for the queries q_1 and q_2 respectively. The page set that users clicked on for the queries q_1 and q_2 may be seen as:

$$\begin{aligned} DC(q_1) &= \{d_{q_{11}}, d_{q_{12}}, \dots, d_{q_{1i}}\} \subseteq D(q_1) \\ DC(q_2) &= \{d_{q_{21}}, d_{q_{22}}, \dots, d_{q_{2j}}\} \subseteq D(q_2) \end{aligned}$$

Similarity based on clickthroughs follows the following principle: If $DC(q_1) \cap DC(q_2) = \{d_1, d_2, \dots, d_k\} \neq \emptyset$, then web pages d_1, d_2, \dots, d_k represent the common topics of queries q_1 and q_2 . The similarity between the queries q_1 and q_2 is determined by $DC(q_1) \cap DC(q_2)$. Therefore, the similarity between the two queries is proportional to the shared number of clicked (or selected) web pages, taken individually, as follows:

$$similarity_{naive-session}(q_1, q_2) = \frac{DN(q_1, q_2)}{\max(dn(q_1), dn(q_2))} \quad (4)$$

where $dn(\cdot)$ is the number of clicked pages for a query, $DN(q_1, q_2)$ is the number of page clicks in common.

In spite of its simplicity, this measure demonstrates a surprising ability to cluster semantically related queries that contain different words. In Figure 1, "Atomic bomb", "Nagasaki", and "Nuclear weapon" are clustered as similar queries since all of them correspond to a clicked web page about atomic bomb. It is obvious that no approach that uses only the keywords in the queries could create such a cluster since these queries are made up of different keywords.

In addition, this measure is also very useful in distinguishing between queries that happen to be worded similarly but stem from different information needs. For example, in Figure 1, one user asked for "java" and clicked

on the web pages about tour in Java Island, one user asked "java" and clicked the web pages about Java programming language, and another user asked the same "java" query and clicked the web pages about Java coffee. These different search intentions can be easily distinguished through user clicks. But the content-based method will certainly consider all of them as the same query. This kind of distinction can be further exploited for sense disambiguation in a user interface.

3.3 Combination of Query Clustering and Document Clustering

The above similarity function is a naive one and only effective when the clicked web pages are constrained in a relatively narrow range and many users click the same web pages. Especially, when two queries leading to two separate but similar page clicks, the similarity between these two queries cannot be detected. In Figure 1, "Manhattan Project" is not detected as a similar query to "atomic bomb" because its related web pages are about "US history: Manhattan Project", although they are very close to the "atomic bomb" web page but not exactly the same.

To overcome this difficulty and obtain query clusters with broader cov-

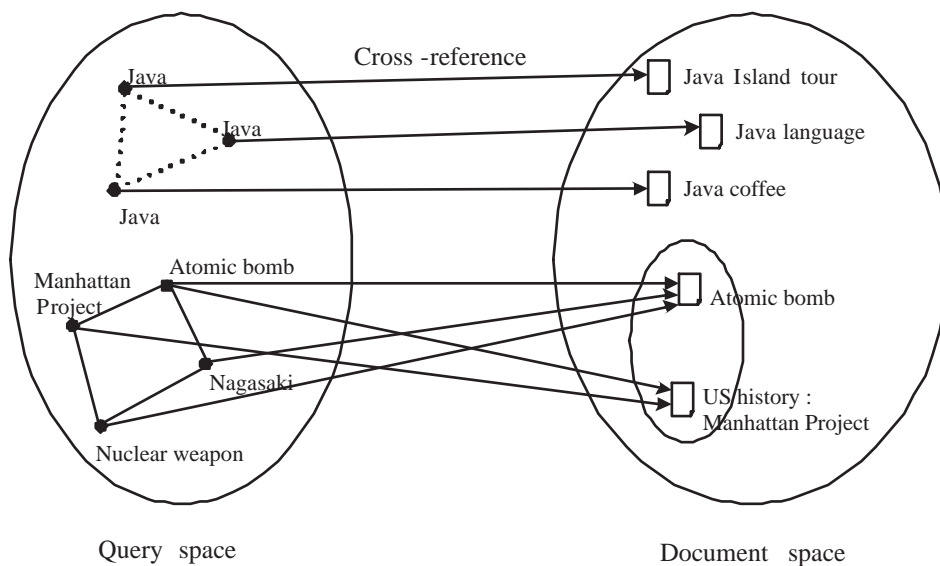


Figure 1: Session-based Query Clustering

erage, document clustering can be used to group the similar web pages first, and then two queries pointing to two different web pages in the same cluster can be recognized as similar queries. There are two ways to achieve such a kind of combination of query clustering and document clustering — unidirectional clustering and bidirectional clustering. In unidirectional clustering, documents need to be collected and clustered ahead of time, and then the document clusters are used to calculate the similarities between queries. In this case, the document clustering and query clustering are two separate phases. On the contrary, bidirectional clustering employs an iterative, agglomerative method to cluster queries and documents simultaneously.

3.3.1 Unidirectional Clustering

Document clustering has been investigated for use in a number of different areas of text mining and information retrieval for decades. Numerous document clustering algorithms have appeared in the literature [30]. In recent years, document clustering was also extended to organizing the web pages returned by a search engine [32].

There are two ways to take advantage of document clustering. One of them is relatively straightforward. First, all web pages are scattered into n different clusters C_1, C_2, \dots, C_n , using any document clustering algorithm. For each query session, all clicked web pages are substituted by the clusters they belong to. Thus, the similarity between two queries is proportional to their shared number of clusters. Then Formula (4) is used to calculate the similarity value between the two queries, except that $DN(q_1, q_2)$ is the number of clicked clusters in common. For the example in Figure 1, if web pages "atomic bomb" and "US history: Manhattan Project" are grouped into a cluster, queries "atomic bomb" and "Manhattan Project" will be considered as leading to the same clicks and put into the same cluster.

The other way to make use of document clustering is to directly incorporate document similarity measure into the calculation of query similarity. Let $s(d_i, d_j)$ be the function to calculate the similarity between documents, $d_i (1 \leq i \leq m)$ and $d_j (1 \leq j \leq n)$ be the clicked documents for queries q_1 and q_2 respectively. The incorporated similarity function is defined as follows:

$$similarity_{session+doc}(q_1, q_2) = \frac{1}{2} \times \left(\frac{\sum_{i=1}^m (max_{j=1}^n s(d_i, d_j))}{m} + \frac{\sum_{j=1}^n (max_{i=1}^m s(d_i, d_j))}{n} \right) \quad (5)$$

For the same example in Figure 1, if the similarity value between the

two web pages "atomic bomb" and "US history: Manhattan Project" is 0.5, and they are the only two web pages selected for the two queries, then the similarity value between the queries is also 0.5 according to Formula (5). In contrast, their similarity value based on Formula (4) using common clicks is 0. Hence, this new similarity function can recognize a wider range of similar queries.

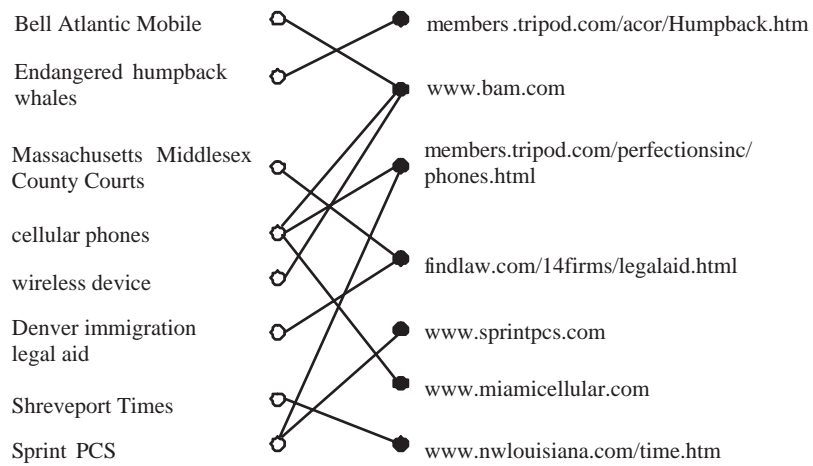
3.3.2 Bidirectional Clustering

As illustrated above, document clustering may effectively enhance query clustering unilaterally. But clustering web pages by content may require storing and manipulating quite a large amount of data, and the unidirectional strategy is only effective when the web pages are constrained in a relatively narrow range. Therefore, cross-reference between queries and web pages is also used to cluster similar documents in a "content-ignorance" manner, that is, *if many people select a common set of documents for the same query, this is a strong indication that this set of documents is similar*. This is a dual hypothesis of the session-based query clustering hypothesis, and has been used in several existing search engines. For example, in CiteSeer (<http://citeseer.nj.nec.com>), once a document is identified, the system also indicates a set of other documents that other people have accessed together with that document. This is a form of implicit document clustering based on cross-references using the above hypothesis. Amazon (<http://amazon.com>) exploits the same hypothesis to suggest books similar to those selected by users.

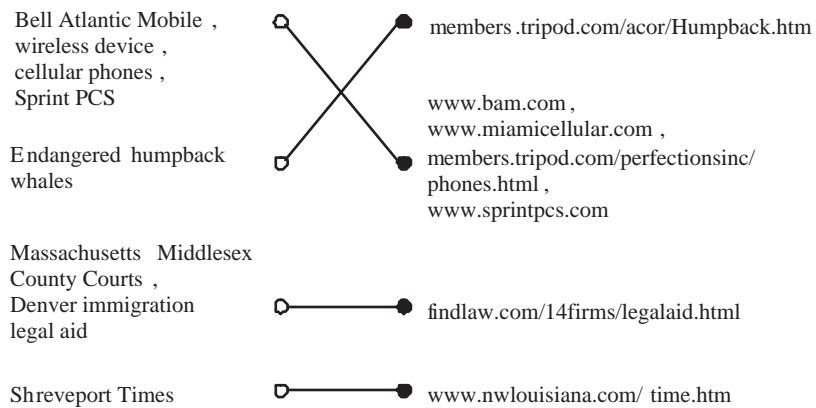
In summary, through query sessions, the distance between two queries can be evaluated by examining their corresponding clicked web pages, and the distance between two web pages can be evaluated by examining their related queries. Therefore, query clustering and (web) document clustering procedures can be combined and reinforced each other to construct a bidirectional clustering method. The intuition of the bidirectional clustering is that similarity between documents can be transferred to queries through their cross-references, and vice versa. Below we discuss a bidirectional clustering algorithm introduced in [1].

The work in [1] is conducted on the Lycos website. First, based on query sessions in web logs, a bipartite graph is constructed where the vertices on one side correspond to unique queries, the vertices on the other side to unique web pages, and edges exist between a query and a web page vertex when they co-occurred in a query session. Figure 2(a) is a sample bipartite graph. Second, a similarity function is defined to measure the similarity between

vertices x and y , both for query vertices and web page vertices. Then an agglomerative clustering algorithm is applied to cluster the queries and web pages respectively. The iterations of these two agglomerative clustering procedures are not independent but interleave each other — first combing the two most similar queries, then combining the two most similar web pages, and repeat the iterations until the resulting graph consists of connected components, each with a single query and web page vertex. Figure 2(b) is the graph at the termination state.



(a) Original Bipartite Graph



(b) Graph at the Termination State

Figure 2: Bidirectional Clustering

This bidirectional clustering strategy is very suitable for clustering queries and web pages simultaneously on general-domain search engine logs with sparse clickthroughs on a huge collection of diverse web pages. The "content-ignorance" characteristic is very valuable when dealing with large datasets, dynamic web pages, text-free pages and pages with restricted access. But while reinforcing query clustering and document clustering each other, bidirectional clustering is also likely to accumulate noises and outliers for both sides. Another unsolved problem is that it is hard to determine the termination condition in advance and the condition in [1] is obviously too severe. Moreover, while exploiting cross-references, [1] rejects the use of the contents of queries and documents. They consider that keyword similarity is totally unreliable. We believe that content words provide some useful information for query clustering that is complementary to cross-references.

3.4 Clustering Queries through Query Co-occurrences

In this part, we briefly introduce another viewpoint about query session and query clustering. As stated in [12], a series of related queries of an individual users can be extracted from web logs and to form a new kind of query session, as shown in Table 2. The session in Table 2 illustrates a typical search scenario: the user first submitted query "WW2" to the search engine and looked for information about the World War Two. Then he/she refined his/her query to "second war" (maybe due to not finding satisfactory answers). Finally, the user submitted the query "world war 2" and stopped this search session. From this scenario we can see that the query co-occurrences in a session can be taken as a useful evidence to judge the correlations among queries [12][8].

Table 2: Another Kind of Query Session

IP address	Cookie	Timestamp	URL
999.999.999.2	BCA8D4C3778E	5/12/00:6:23:21	http://encarta.msn.com/ find/search.asp?s=WW2
999.999.999.2	BCA8D4C3778E	5/12/00:6:23:52	http://encarta.msn.com/ find/search.asp?s=second+war
999.999.999.2	BCA8D4C3778E	5/12/00:6:28:06	http://encarta.msn.com/ find/search.asp?s=world+war+2

Although the above example happens frequently in web logs, another common scenario is that consecutive queries from a user are independent

from each other and the user actually changed his/her interests during the search process. The main challenge is that it's very hard to partition the query log into query sessions so that each query session really corresponds to the query terms submitted by a user for a single information need. Another problem is how to measure the relevance between each pair of query terms in a session.

4 Combining Evidences from Query Content and Query Session

As we mentioned before, content-based similarity function alone is not sufficient to cluster semantically related queries. The main reason is that keywords and meanings do not strictly correspond. The same keyword does not always represent the same information need (e.g. the word "Java" may refer to "Java" the island, coffee, or a programming language); and different keywords may refer to the same concept (e.g. all the terms "atomic bomb", "Manhattan project", "nuclear weapon" and "Nagasaki" are related to the concept "atomic bomb"). Therefore, calculated similarity does not necessarily reflect semantic similarity. This is particularly the case for web queries which typically are very short (usually less than two words [29]).

On the other hand, one may reject the use of query sessions as a reliable criterion, too. In fact, a web page click is not a relevance judgment. Users may well click on irrelevant pages. Therefore, web page clicks, if considered as relevance judgments, are noisy. In addition, a web page can describe more than one specific topic, and can correspond to different queries. Putting together the queries corresponding to the same web page may result in a cluster containing very different queries. These facts may raise reasonable doubt on the reliability of web page clicks as a clustering criterion.

While it is true that relations made by a small number of users are likely to be unreliable, the large amount of information available in web logs makes this less of a problem; we assume that users are more consistent in their relations of relevant web pages than irrelevant ones. Cross-references confirmed by many users are therefore taken to be reliable. One more supporting evidence is the results of pseudo-relevance feedback in IR [31] which show that in reformulating queries using the top-ranked documents as if they are relevant, the retrieval effectiveness can be increased significantly. This shows that useful information about the connection between queries and documents in IR is not restricted to explicit relevance judgments. Even very coarse and noisy indications can help. Our situation is better than

in pseudo-relevance feedback — not only are the documents presented to the user the top-ranked documents, but there is also a further selection by the user. So document clicks are more reliable indication than that used in pseudo-relevance feedback. Therefore, we can only expect better results.

Examinations of the typical web logs showed that most of the queries are followed by only one or two web page selections. People may doubt whether such a small number of clicks can provide a reliable basis for query clustering. This is very similar to the "short-query" problem. Both content words and web page clicks can partially reflect the semantics of the queries. But because of the small numbers of keywords and document clicks, neither of them is sufficient to be used alone. Therefore, some combined measures should be defined to take advantage of both strategies. A simple way to do this is to combine both measures linearly, as follows:

$$similarity = \alpha \times similarity_{content-based} + \beta \times similarity_{session-based} \quad (6)$$

Some experiments [29] show that the combined measure can be used to cluster more queries in a more precise way. The combined measure raises the question of how to set parameters α and β . We believe that these parameters should be set according to the editor's objectives.

5 Choosing a Clustering Algorithm

Besides similarity functions, choosing an appropriate clustering algorithm is also very critical to the effectiveness and efficiency of the query clustering process. As to choosing web query clustering algorithm, there are several rules that should be followed. First, as query logs usually are very large, the algorithm should be capable of handling a large data set within reasonable time and space constraints. Second, due to the fact that the log data changes daily, the algorithm should be easily extended to cluster new queries incrementally. Third, the algorithm should not require manual setting of the resulting form of the clusters, e.g. the number or the maximal size of clusters. It is unreasonable to determine these parameters in advance. Finally, since we only want to find FAQs, the algorithm should filter out those queries with low frequencies.

There is a great deal of work done previously in clustering algorithms. Basically, they can be divided into two groups of clustering algorithms: non-hierarchical and hierarchical. K-Means and Agglomerative Hierarchical Clustering (AHC) are representatives of these two groups [5].

K-Means constructs a partition of n points into the final required k clusters. First, k clusters is chosen such that the points are mutually farthest apart. Next, it examines each point and assigns it to one of the clusters depending on the minimum distance. The centroid of a cluster is recalculated every time a point is added to it and this continues until all the points are grouped into the final required number of clusters. K-Means has a time complexity which is linear in the number of points, which complies with the speed requirement of clustering a large amount of queries. Its main disadvantage is that it is known to be the most effective when the desired clusters are approximately spherical with respect to the similarity measure used. There is no reason to believe that queries should fall into approximately spherical clusters. Another disadvantage of K-Means is that the number of clusters, k , needs to be determined at the onset, which is usually unrealistic.

Agglomerative Hierarchical Clustering is probably the most commonly used clustering algorithm and it creates a hierarchical decomposition of a population of points. In AHC, each object is initially placed into its own group. Therefore, if we have n objects to cluster, we start with n groups. Each of these groups contains only a single object and is known as a singleton. If the distance between a closest pair of groups is less than the threshold distance, these groups become linked and are merged into a single group. This merge procedure is repeated until there is no other merge possible. To run an agglomerative clustering, a measure should be provided to calculate the distance between two objects. In addition, another measure is needed to determine which groups should be linked. Some options are single-link, group-average, complete-link, etc. AHC is typically slow when applied to large dataset. Single-link and group-average methods typically take $O(n^2)$ time, while complete-link methods typically take $O(n^3)$ time. Although it is not required to know the number of final clusters ahead of time, the halting criteria of AHC are arbitrary and its performance is poor in domains with many outliers.

The density-based clustering method, DBSCAN [6] and its incremental version, Incremental DBSCAN [7] are good candidate algorithms which satisfy all of the above requirements. DBSCAN does not require the number of clusters as an aprior input parameter. Density-based methods are based on the idea that it is likely that in a space of objects, dense objects should be grouped together into one cluster. Thus, a cluster is a region that has a higher density of points than its surrounding region. A cluster consists of at least the minimum number of points — *MinPts* (to eliminate very small clusters as noise); and for every point in the cluster, there exists another point in the same cluster whose distance is less than the distance threshold

Eps (points are densely located). DBSCAN is a one pass algorithm since it groups neighboring objects of the database into clusters based on a local cluster condition. It is very efficient if the retrieval of the neighborhood of an object is efficiently supported by some indexing mechanism. For the query clustering case, an inverted file is very efficient to search for neighbor points. All clusters consisting of less than the minimum number of points are considered as "noise" and are discarded. Thus those queries with low frequencies will be filter out. Previous experiments showed that DBSCAN outperforms CLARANS [18] by a factor of between 250 and 1900, which increases with the size of the data set. The ability of Incremental DBSCAN to update incrementally is due to the density-based nature of the DBSCAN method, in which the insertion or deletion of an object only affects the neighborhood of this object. In addition, based on the formal definition of clusters, it has been proven that the incremental algorithm yields the same results as DBSCAN. Performance evaluations show that Incremental DBSCAN is more efficient than the basic DBSCAN algorithm.

6 A Case Study on Encarta Online Search Engine

To test various query clustering methods, we carried out a study on the Encarta encyclopedia (<http://encarta.msn.com>), which can be accessed on the Web. The Encarta editors have an impending need of determining the FAQs of the users. This research work is launched specifically to answer the need of Encarta editors.

Encarta contains a large number of documents organized in a hierarchy. A document (or article) is written on a specific topic, for instance, about a country. In contrast with other documents on the Web, the quality of the documents is well controlled. The current search engine used on Encarta uses some advanced strategies to retrieve relevant documents using keywords. The answer to a query/question is a set of documents. Although the search engine suffers from the same common problems of search engines using keywords, the result list does provide a good indication of the document contents. Therefore, when a user clicks on one of the documents in the result list, he/she has some idea about what can be found in the document. This provides us with a solid basis for using user clicks as a valid indication of relevance.

Encarta online is not a static encyclopedia as the printed one. It evolves in time. In fact, a number of human editors and indexers are working on the improvement of the encyclopedia so that users can find more information

from Encarta and in a more precise way. Human editors work on expanding contents by adding new documents according to users' "hot topics". And human indexers want to increase search performance by adding new index entries or adjusting index term weights according to users' "hot query terms". Moreover, if many users asked the same questions (FAQ) in a certain period of time, then the answers to these questions will be checked manually and directly linked to the questions.

For a long time, all of these tasks are done based on manual analysis of user queries in the logs. Our goal is to develop a clustering system that helps the editors quickly identify the FAQs. The key problem is to determine a similarity function which will enable the system to form clusters of truly similar queries.

6.1 Query Sessions

We collected one-month user logs (about 22 GB) from the Encarta web site. From these logs we extracted more than two millions user query sessions. A query session is defined as: $session = query \rightarrow [clicked-document]^*$. Each session corresponds to one query and the Encarta documents the user clicked on. 49% of queries contain only one keyword and 33% of queries contain two keywords. The average length of all queries is 1.86. The distribution of query length is similar to those reported by others. Because the number of queries is too big to conduct detailed evaluations, we randomly chose 20,000 query sessions from them for our study.

In Encarta, once a user query is input, a list of documents is presented to the user, together with the document titles and snippets. We observed that about half of query sessions have document clicks. Out of these sessions, about 90% have 1-2 document clicks. Because the document titles and snippets in Encarta are carefully chosen, they give the user a good idea of the contents of the documents. Therefore, if a user clicks on a document, it is likely that the document is relevant to the query, or at least related to it to some extent. Even if some of the document clicks are erroneous, we can expect that most users do click on relevant/strongly related documents.

6.2 Document Collection and Document Similarity

The Encarta documents collection is made up of 41,942 documents in various topics. Those documents are not isolated; they are organized into a hierarchy that corresponds to a concept space. The hierarchy contains 4 levels. The first level is the root. The second level contains 9 categories,

such as "physical science & technology", "life science", "geography", etc. These categories are divided into 93 subcategories. The last level (the leaf nodes) is made up of tens of thousands of documents. Roughly, this document hierarchy corresponds to a domain/concept hierarchy. Instead of using document contents, we use the document hierarchy to calculate conceptual distances between documents. This distance is determined as follows: the lower the common parent node the two documents have, the shorter the conceptual distance between the two documents is, and the higher their conceptual similarity is. Let $F(d_i, d_j)$ denote the lowest common parent node for documents d_i and d_j , $L(x)$ the level of node x , $L-Total$ the total levels in the hierarchy (i.e. 4 for Encarta). The conceptual similarity between two documents is defined as follows:

$$s(d_i, d_j) = \frac{L(F(d_i, d_j)) - 1}{L-Total - 1} \quad (7)$$

In particular, $s(d_i, d_i) = 1$; and $s(d_i, d_j) = 0$ if $F(d_i, d_j) = root$.

We incorporate this document similarity measure into Formula (5) and use it as a new similarity measure for queries.

6.3 Evaluation of Different Similarity Functions

To test how different similarity functions affect the query clustering results, we conducted a series of experiments using Encarta logs and document collection. Specifically, we tested the following four similarity functions on the extracted 20,000 query sessions:

- Query keyword similarity (*K-Sim*, *Formula (1)*),
- Naive similarity using clickthroughs (*S-Sim*, *Formula (4)*),
- Query keyword + Naive clickthrough similarity (*K+S-Sim*, *Formula (1)+(4)*), and
- Query keyword + clickthrough + document similarity (*K+S+D-Sim*, *Formula (1)+(5)+(7)*).

DBSCAN is chosen as the clustering algorithm. The minimal density parameter (*MinPts*) was set to 3 uniformly, which means that only those clusters containing at least 3 queries are kept. Then we varied the similarity threshold ($= 1 - Eps$) from 0.5 to 1.0. As for the combined measures, we assigned weight 0.5 to both α and β uniformly.

Like other applications using clustering algorithms, there is no standard methodology to evaluate the quality of the results of query clustering. We borrow two IR metrics as our quality measures:

- Precision — the ratio of the number of similar queries to the total number of queries in a cluster.
- Recall — the ratio of the number of similar queries to the total number of all similar queries for these queries (those in the current cluster and others).

For every similarity function, we randomly selected 100 clusters. For each cluster, we calculated precision by manually checking the queries. Since we do not know the actual intentions of users with their queries, we simply guess, taking into account both queries and clicked documents. We report the average precision of the 100 clusters in Figure 3, where all the four functions are shown with similarity threshold varying from 0.5 to 1.0.

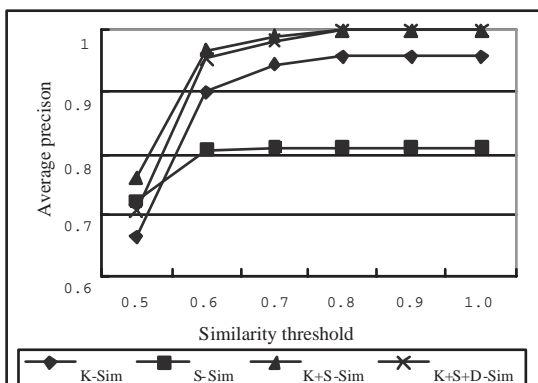


Figure 3: Precision for Four Kinds of Similarity Functions

We first observe that the combinations of keywords and clickthroughs in query sessions ($K+S-Sim$ and $K+S+D-Sim$) result in higher precision than the two criteria separately. When similarity threshold is equal to or higher than 0.6, $K+S-Sim$ and $K+S+D-Sim$ have very high precision (above 95%). When the similarity threshold is higher than 0.8, the precision for both similarity functions reaches 1.0.

For clustering using single evidence, we observe that the highest precision that can be reached by $K-Sim$ is about 96%, when all queries in a cluster contain identical keywords. This means the ambiguity of keywords

will only bring in about 4% errors. This number is much lower than our expectation. A possible reason is that users usually are aware of word ambiguity and would like to use more precise queries. For example, instead of using "Java", users use "Java Island" or "Java programming language" to avoid ambiguities.

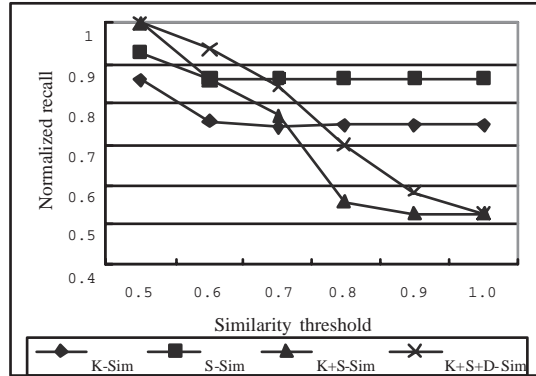


Figure 4: Recall for Four Kinds of Similarity Functions

It is difficult to use the recall metric directly for clustering because no standard clusters or classes are available. Therefore, we use a different measure to reflect, to some extent, the recall factor: normalized recall. This factor is calculated as follows:

- For any similarity function, we collect the number of correctly clustered queries in all the 100 clusters. This is equal to the total number of clustered queries times the precision.
- Then we normalize this value by dividing it with the maximum number of correctly clustered queries. In our case, this number is 12357 which is obtained by *K+S+D-Sim* when the similarity threshold is 0.5. This normalization aims at obtaining a number in the range $[0, 1]$.

Figure 4 shows the normalized recalls for the four similarity functions when similarity threshold varies from 0.5 to 1.0. We observe that, when similarity threshold is below 0.6, *K+S+D-Sim* and *K+S-Sim* result in better normalized recall ratios than using two other functions on single evidence. This shows that both functions can take advantage of evidences from query and session. However, when similarity threshold increases, then normalized

recall ratios drop quickly. On the other hand, there is almost no change for *S-Sim* and *K-Sim* for threshold higher than 0.6. Again, this is due to the small number of keywords per query and document clicks per query.

Although the precision of *K+S+D-Sim* is very close to *K+S-Sim* (Figure 3), the normalized recall of *K+S+D-Sim* is always higher than *K+S-Sim* with a margin of about 10% (Figure 4). This shows that, when combined with keywords, the consideration of document similarity is helpful for increasing recall significantly without decreasing precision.

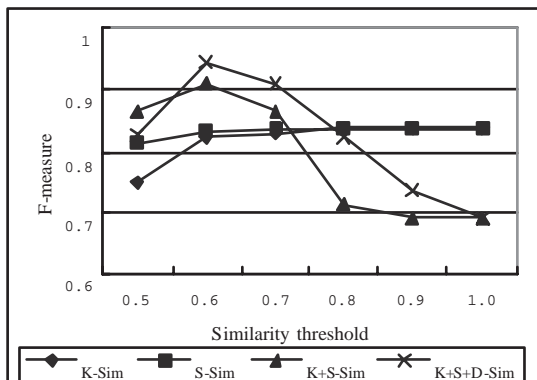


Figure 5: F-measures for Four Kinds of Similarity Functions

In order to compare the global quality of the four functions, we use the F-measure [16] as metric (in which the recall ratio is replaced by our normalized recall). Although the modified F-measure is different from the traditional one, it does provide some indication on the global quality of different similarity functions. Figure 5 shows this evaluation. We can see that within the threshold range of [0.5, 0.7], *K+S+D-Sim* and *K+S-Sim* are better than the single-evidence functions. In particular, when similarity threshold is equal to 0.6, *K+S+D-Sim* reaches the highest F-measure value (0.94).

All the above experiments show that it is beneficial to combine evidences from query contents, query sessions, and documents in query clustering.

7 Summary

Query clustering is a class of techniques developed in recent years to meet the requirements of more sophisticated web search. Query clustering is not

an easy task. It requires a proper estimation of query similarity. Unlike traditional document clustering, due to the short length and word ambiguity of user queries, the similarity between queries cannot be accurately measured only based on query contents (e.g., keywords, query forms, etc.) and, thus, those really semantically related queries cannot be grouped together effectively.

Query session is an extension to query and, accordingly, query clustering also is changed to "query session clustering". A query session is made up of the query and the subsequent activities the user performed. From accumulated web logs, a large amount of query sessions can be extracted using some user identification and session identification mechanisms. The click-through information in query sessions provides very valuable evidences to deduce users' search intentions more precisely than solely relying on query content. Although some naive methods based on clickthroughs in query sessions can improve the clustering results significantly, a more effective way is to combine query clustering and document clustering through query sessions. There are two ways to achieve such a combination — unidirectional clustering and bidirectional clustering. Unidirectional clustering utilizes document clustering to enhance query clustering unilaterally. On the contrary, bidirectional clustering employs an iterative, agglomerative method to cluster queries and documents simultaneously.

Content-based and session-based query clustering represent two different points of view and have their own advantages and shortcomings. Therefore, some combined measures taking advantage of both criteria may be more appealing. How to choose proper clustering algorithm in the query clustering context is also discussed, and four principal rules guiding the choice decision are given. Finally, a case study on Encarta web site is introduced in detail. The characteristics of different clustering methods are demonstrated and evaluated.

References

- [1] D. Beeferman and A. Berger, Agglomerative clustering of a search engine query log, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2000) pp. 407-416.
- [2] R. Cooley, B. Mobasher, and J. Srivastava, Data preparation for mining World Wide Web browsing patterns, *Journal of Knowledge and Information Systems* Vol.1 No.1 (1999).

- [3] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, Probabilistic query expansion using query logs, *Proceedings of the Eleventh World Wide Web conference (WWW 2002)* (2002) pp. 325-332.
- [4] E. De Lima and J. Pedersen, Phrases recognition and expansion for short, precision-biased queries based on a query log, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1999) pp. 145-152.
- [5] R. C. Dubes and A. K. Jain, *Algorithms for Clustering Data*, (Prentice Hill, 1988).
- [6] M. Ester, H. Kriegel, J. Sander, and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (1996) pp. 226-231.
- [7] M. Ester, H. Kriegel, J. Sander, M. Wimmer, and X. Xu, Incremental clustering for mining in a data warehousing environment, *Proceedings of the 24th International Conference on Very Large Data Bases* (1998) pp. 323-333.
- [8] L. Fitzpatrick. and M. Dent, Automatic feedback using past queries: social searching? *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1997) pp. 306-312.
- [9] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais, The vocabulary problem in human-system communication, *CACM* Vol.30 No.11 (1987) pp. 964-971.
- [10] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Part III "Inexact Matching, Sequence Alignment, and Dynamic Programming"*, (Press of Cambridge University, 1997).
- [11] M.H. Hansen and E. Shriver, Using navigation data to improve IR functions in the context of web search, *Proceedings of the 10th International Conference on Information and Knowledge Management (ACM CIKM 2001)*, (2001) pp. 135-142.
- [12] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang, Query-session-based term suggestion for interactive web search, *WWW10 Poster Proceedings* (2001).

- [13] K. Kukich, Techniques for automatically correcting words in text, *ACM Computing Surveys* Vol.24 No.4 (1992) pp. 377-439.
- [14] V.A. Kulyukin, K.J. Hammond, and R.D. Burke, Answering questions for an organization online, *Proceedings of AAAI'98* (1998) pp. 532-538.
- [15] D.D. Lewis and W.B. Croft, Term clustering of syntactic phrases, *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1990) pp. 385-404.
- [16] Z. Lu and K. McKinley, Partial collection replication versus caching for information retrieval systems, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2000) pp. 248-255.
- [17] G.A. Miller (eds.), WordNet: an on-line Lexical Database, *International Journal of Lexicography* Vol.3 No.4 (1990).
- [18] R. Ng and J. Han, Efficient and effective clustering method for spatial data mining, *Proceedings of the 20th International Conference on Very Large Data Bases* (1994) pp. 144-155.
- [19] P. Pirolli, J. Pitkow, and R. Rao, Silk from a sow's ear: Extracting usable structures from the Web. *Proceedings of 1996 Conference on Human Factors in Computing Systems (CHI-96)* (1996).
- [20] J. Pitkow, In search of reliable usage data on the WWW, *Proceedings of the Sixth World Wide Web conference (WWW6)* (1997) pp. 451-463.
- [21] M. Porter, An algorithm for suffix stripping, *Program* Vol.14 No.3 (1980) pp. 130-137.
- [22] J. Rocchio, Relevance feedback in information retrieval, in G. Salton (eds.) *The Smart Retrieval System — Experiments in Automatic Document Processing* (Prentice-Hall Englewood Cliffs, 1971) pp. 313-323.
- [23] G. Salton and C. Buckley, Improving retrieval performance by relevance feedback, *Journal of the American Society for Information Science*, Vol.41 No.4 (1990) pp. 288-297.
- [24] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval* (McGraw-Hill Book Company, 1983).
- [25] R. Srihari and W. Li, Question answering supported by information extraction, *Proceedings of TREC8* (1999) pp. 75-85.

- [26] C.J. van Rijsbergen, *Information Retrieval (Second Edition)* (Butterworths, London, 1979).
- [27] E. Voorhees, N.K. Gupta, and B. Johnson-Laird, Learning collection fusion strategies, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1995) pp. 172-179.
- [28] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, Clustering user queries of a search engine, *Proceedings of the Tenth World Wide Web conference (WWW10)* (2001) pp. 162-168.
- [29] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, Query Clustering Using User Logs, *ACM Transactions on Information Systems (ACM TOIS)* Vol.20 No.1 (2002) pp. 59-81.
- [30] P. Willett, Recent trends in hierarchical document clustering: A critical review, *Information Processing and Management* Vol.24 No.5 (1988) pp. 577-597.
- [31] J. Xu and W.B. Croft, Query expansion using local and global document analysis, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996) pp. 4-11.
- [32] O. Zamir and O. Etzioni, Web document clustering: A feasibility demonstration, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1998).