Answering Relationship Queries on the Web

Gang Luo   Chunqiang Tang   Ying-li Tian
IBM T.J. Watson Research Center
{luog, ctang, yltian}@us.ibm.com

ABSTRACT

Finding relationships between entities on the Web, e.g., the connections between different places or the commonalities of people, is a novel and challenging problem. Existing Web search engines excel in keyword matching and document ranking, but they cannot well handle many relationship queries. This paper proposes a new method for answering relationship queries on two entities. Our method first respectively retrieves the top Web pages for either entity from a Web search engine. It then matches these Web pages and generates an ordered list of Web page pairs. Each Web page pair consists of one Web page for either entity. The top ranked Web page pairs are likely to contain the relationships between the two entities. One main challenge in the ranking process is to effectively filter out the large amount of noise in the Web pages without losing much useful information. To achieve this, our method assigns appropriate weights to terms in Web pages and intelligently identifies the potential connecting terms that capture the relationships between the two entities. Only those top potential connecting terms with large weights are used to rank Web page pairs. Finally, the top ranked Web page pairs are presented to the searcher. For each such pair, the query terms and the top potential connecting terms are properly highlighted so that the relationships between the two entities can be easily identified. We implemented a prototype on top of the Google search engine and evaluated it under a wide variety of query scenarios. The experimental results show that our method is effective at finding important relationships with low overhead.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: search process

General Terms: Algorithms, Experimentation

Keywords: relationship query, web search

1. INTRODUCTION

A *relationship query* (RQ) asks for the relationships between two or more entities [10], e.g., the connections between different places or the commonalities of people. As mentioned in [14], answers to RQs are useful for government and military intelligence analysts, news reporters, financial industry analysts, historians, biographers, lawyers, and detectives. These answers can even help high school students write homework essays. To encourage research on answering RQs, the Advanced Research and Development Activity (ARDA) sponsored an Advanced Question Answering for Intelligence (AQUAINT) program [2] that includes a relationship pilot for RQs. In the 2005 Text REtrieval Conference (TREC), a new relationship task was added into the Question Answering (QA) track [15]. However, all the previous efforts focus on relatively small, traditional document sets and no attempt has been made to answer RQs on the Web – the largest knowledge base in the world that contains both much more information and much more noise than traditional document sets [10, 24].

Web searchers are often interested in finding relationships between entities. For example, Mr. Glenn Klausman is a Florida attorney practicing personal injury law. Suppose Glenn plans to attend a party. He notices that Dr. John Robert Schrieffer, a professor at Florida State University, is also invited to the party. By searching on the Web (e.g., typing John's name in Google and reading the returned first result page), Glenn finds out that John is a Nobel Prize laureate in physics and would like to chat with him. (John co-invented the BCS-theory of superconductivity.) To get prepared, Glenn does some background search on the Web, attempting to find some relationship between him and John. However, this is not an easy task. Since John is a world renowned physicist, most Web pages about John are related to physics, about which Glenn does not have much idea. Especially, this is true for the top several Web pages returned by a Web search engine when using John's name as query keywords. Also, typing both John's name and Glenn's name in a Web search engine does not help, as there is no Web page that mentions both names. In general, existing Web search engines excel in keyword matching but they have no good support for this type of RQs.

Actually, a relationship does exist between Glenn and John. On Sep. 24, 2004, John ran into a car accident, in which several people were injured (see Sections 2 and 3 for the story). Recall that Glenn's expertise is in personal injury law. If Glenn can find this relationship, he may offer some help to John when they meet at the party.

In this paper, we treat the Web as a huge database and attempt to find relationships from unstructured documents. We focus on the most important RQs that ask for the relationships between two entities E_1 and E_2. For a RQ, if some Web pages mention both entities and their relationship, finding this relationship may be an easy task – typing both entities' keywords in a Web search engine often leads to those Web pages. Alternatively, if the top few Web pages retrieved from a Web search engine for either entity contain this relationship, the searcher may also be able to find this relationship quickly. However, if neither of these is the case, which is not uncommon [10], existing Web search engines cannot help much in finding the answer. Unless the searcher can guess the relationship, he usually has to spend a lot of time reading many Web pages related to either entity, hoping that he can manually discover the relationship.

To address this problem, we propose a new method for answering RQs on the Web. Our method matches Web pages for entity E_1 and Web pages for entity E_2 that are retrieved from a Web search engine, and then ranks the matched Web page pairs. A Web page pair consists of one Web page for either entity. The top ranked Web page pairs are likely to contain the relationships between E_1 and E_2. In the ranking process, one main challenge is to effectively filter out the large amount of "noise" (i.e., irrelevant information) in the Web pages without losing much useful information. To achieve this, windowing around query keywords is first used to remove some noise from these Web pages. Then for each pair of the Web pages, we identify the potential *connecting*

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

terms that capture the relationship between E_1 and E_2 and compute term weights based on the characteristics of the two Web page sets for the two entities. As a new filtering technique that can reduce noise from long Web pages, all potential connecting terms are sorted in descending order of their term weights and only the top potential connecting terms are used to compute the similarity value between this pair of Web pages. The computed similarity value roughly reflects the likelihood that this pair of Web pages mention some relationship between E_1 and E_2. Finally, the Web page pairs are sorted in descending order of their similarity values and the top Web page pairs are returned to the searcher. For each such top Web page pair, both the top potential connecting terms and the query terms are properly highlighted so that the searcher can easily identify the relationships between E_1 and E_2.

To the best of our knowledge, this is the first work on answering RQs on the Web. Our method has the following advantages. It can find multiple relationships between two entities simultaneously. It can find important relationships even in the presence of a lot of "noise." The quality of search results is insensitive to parameter changes. It can be implemented either inside or on top of a Web search engine. In the latter case, it can be implemented on either the client side or the server side. Moreover, its interface is user friendly and searchers can keep using the familiar keyword query interface of Web search engines. These advantages are verified through a prototype implementation of our method on top of the Google search engine.

The rest of the paper is organized as follows. Section 2 presents the details of our method. Section 3 evaluates the effectiveness of our method under a wide variety of query scenarios. We discuss related work in Section 4 and conclude in Section 5.

2. ANSWERING RELATIONSHIP QUERIES

Consider a RQ that asks for the relationships between two entities E_1 and E_2. Because of several reasons, the searcher may not be able to find desired relationships between E_1 and E_2 by using a Web search engine:

(1) The top few Web pages retrieved for E_1 and the top few Web pages retrieved for E_2 do not contain any desired relationship between E_1 and E_2.
(2) No Web page mentions both E_1 and E_2 and their relationship.
(3) By typing in a Web search engine a query that contains all the keywords of E_1 and E_2, the returned top few Web pages do not mention any desired relationship between E_1 and E_2. Note that some of these Web pages may either (a) mention some relationships that are uninteresting to the searcher, or (b) just happen to incidentally mention both E_1 and E_2.
(4) No desired relationship exists between E_1 and E_2.

Our method can work for most situations except (4).

2.1 User Interface

The user interface of our prototype contains two parts: the query interface and the answer interface. Figure 1 shows the query interface. It is similar to the traditional keyword query interface of Web search engines, except that there are two inputs: one for the keywords of entity E_1, and the other for the keywords of entity E_2.

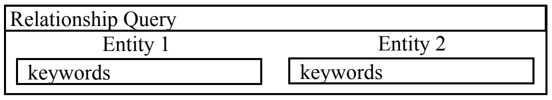
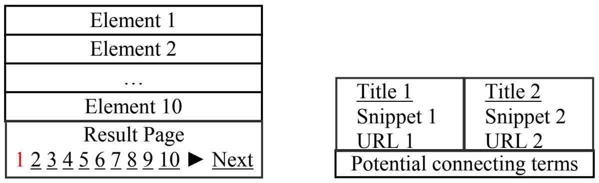


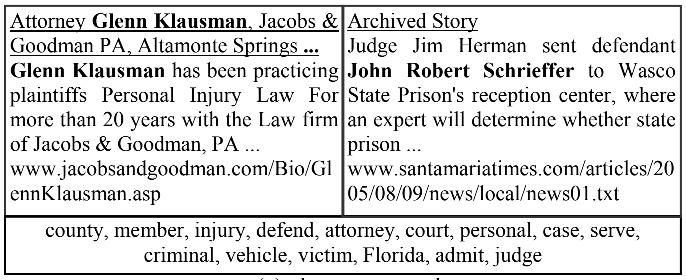
Figure 1. Query interface.

Figure 2 shows the format of answers we would like to provide for RQs – Web page pairs. Answers to a RQ are organized into one or more result pages. Each result page contains ten elements. Each element corresponds to a pair of Web pages (P_1, P_2), where P_i is related to entity E_i (i=1, 2). In our current approach, P_i is one of the top Web pages returned from a Web search engine when using the keywords of E_i as query keywords. (In general, P_i can be any Web page related to E_i, regardless of how P_i is obtained.) All the Web page pairs are sorted in descending order of estimated likelihoods that they contain some relationships between E_1 and E_2.



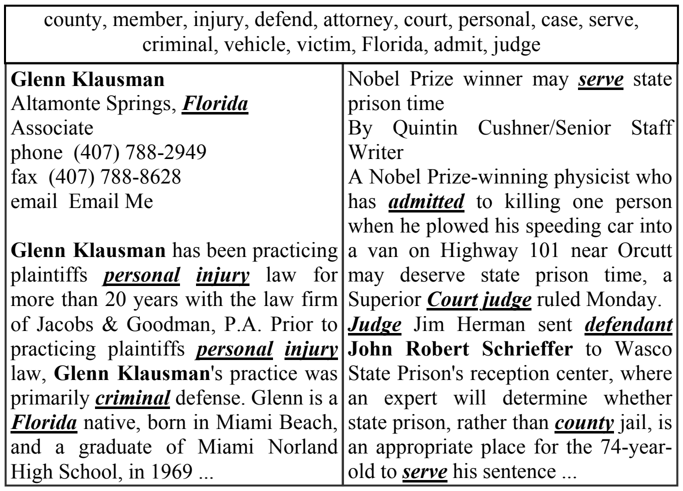
(a) high-level answer format (b) element format

Figure 2. Answer interface.



Attorney **Glenn Klausman**, Jacobs & Goodman PA, Altamonte Springs **...** **Glenn Klausman** has been practicing plaintiffs Personal Injury Law For more than 20 years with the Law firm of Jacobs & Goodman, PA ... www.jacobsandgoodman.com/Bio/GlennKlausman.asp	Archived Story Judge Jim Herman sent defendant **John Robert Schrieffer** to Wasco State Prison's reception center, where an expert will determine whether state prison ... www.santamariatimes.com/articles/2005/08/09/news/local/news01.txt
county, member, injury, defend, attorney, court, personal, case, serve, criminal, vehicle, victim, Florida, admit, judge	

(a) element example



county, member, injury, defend, attorney, court, personal, case, serve, criminal, vehicle, victim, Florida, admit, judge	
Glenn Klausman Altamonte Springs, ***Florida*** Associate phone (407) 788-2949 fax (407) 788-8628 email Email Me **Glenn Klausman** has been practicing plaintiffs ***personal*** ***injury*** law for more than 20 years with the law firm of Jacobs & Goodman, P.A. Prior to practicing plaintiffs ***personal*** ***injury*** law, **Glenn Klausman**'s practice was primarily ***criminal*** defense. Glenn is a ***Florida*** native, born in Miami Beach, and a graduate of Miami Norland High School, in 1969 ...	Nobel Prize winner may ***serve*** state prison time By Quintin Cushner/Senior Staff Writer A Nobel Prize-winning physicist who has ***admitted*** to killing one person when he plowed his speeding car into a van on Highway 101 near Orcutt may deserve state prison time, a Superior ***Court*** ***judge*** ruled Monday. ***Judge*** Jim Herman sent ***defendant*** **John Robert Schrieffer** to Wasco State Prison's reception center, where an expert will determine whether state prison, rather than ***county*** jail, is an appropriate place for the 74-year-old to ***serve*** his sentence ...

(b) element click-through example

Figure 3. Answer example.

For either i (i=1, 2), an element contains the title T_i, the snippet (i.e., some words extracted from Web page P_i), and the URL of P_i. These three parts are returned from a Web search engine (such as Google or Yahoo!) when using the keywords of entity E_i as query keywords. Also, an element contains no more than 15 potential connecting terms, where *connecting terms* are words that capture the relationships between E_1 and E_2. (The reason for choosing the number 15 is discussed in Section 2.6.) These potential connecting terms appear in both P_1 and P_2 and are sorted in descending order of estimated likelihoods that they are real connecting terms. When the searcher clicks title T_i (i=1, 2), P_i is displayed. When the searcher clicks the entire element, P_1 and P_2 are displayed shoulder to shoulder under the list of potential connecting terms. In either case, both the potential connecting terms and the keywords of E_i (i=1, 2) are highlighted in P_i using different highlighting methods. In this way, the searcher can easily discover the relationships between E_1 and E_2. A typical case is that P_1 mentions that E_1 is related to entity E_c, P_2 mentions that E_2 is also related to E_c, and the relationship between E_1 and E_2 is E_c. Figure 3 shows an example of the answer to the RQ that is mentioned in the introduction.

2.2 Overview of Our Approach

Our method can be implemented either inside or on top of a Web search engine. Here our method is implemented on top of a Web search engine. In Section 2.7, we discuss the case of implementing our method inside a Web search engine.

We first obtain Web pages for either entity from a Web search engine (our experiments use Google). Then we match these Web pages and obtain a set of Web page pairs. For each pair of Web pages (P_1, P_2), where P_i is related to entity E_i (i=1, 2), a similarity value is used to reflect the likelihood that (P_1, P_2) mentions some relationship between E_1 and E_2. Our method is implemented in the following steps (find the details of each step in following subsections):

Step 1: For either i (i=1, 2), use the keywords of entity E_i as query keywords and retrieve from the Web search engine the top M_i Web pages. M_1 and M_2 are two parameters of our method. Let S_i (i=1, 2) denote the set of top M_i Web pages for E_i.

Step 2: Pre-process the Web pages in set S_i (i=1, 2) to reduce noise.

Step 3: For each pair of Web pages (P_1, P_2), where $P_1 \in S_1$ and $P_2 \in S_2$, identify the potential connecting terms in (P_1, P_2) and compute the similarity value between P_1 and P_2. Recall that connecting terms capture the relationships between E_1 and E_2.

Step 4: Sort all the Web page pairs in descending order of their similarity values. Return the top Web page pairs to the searcher.

2.3 Step 1: Obtaining Web pages

For either i (i=1, 2), we use the keywords of entity E_i as query keywords and retrieve from the Web search engine the URLs of the top M_i Web pages. (Many Web search engines provide their own APIs [1] for this purpose.) For each URL, the corresponding Web page is retrieved from the Web. The purpose of Step 1 is to obtain the important Web pages related to either E_1 or E_2. Important relationships between E_1 and E_2 are likely to be embedded in some of those Web pages. Later steps attempt to extract these relationships by analyzing those Web pages. Generally, the larger the numbers M_1 and M_2, the more likely the relationships are embedded in some of those Web pages. However, if M_1 and M_2 are too large, due to noise (i.e., irrelevant information) in the Web pages, it would be difficult to identify the right Web page pairs that mention the relationships between E_1 and E_2. Our experiments show that $M_1=M_2=50$ is usually sufficient to discover important relationships between E_1 and E_2. We also provide some interface to allow the searcher to change the default values of M_1 and M_2 if necessary.

2.4 Step 2: Document Pre-processing

Step 2 employs a pre-processing procedure to reduce noise from Web pages. Let K_i (i=1, 2) denote the set of keywords of entity E_i. That is, $K_1 \cup K_2$ represents all *query keywords*. Recall that S_i (i=1, 2) denotes the set of top M_i Web pages for E_i. For each Web page P in set S_i (i=1, 2), the following operations are performed:

Operation 1: All HTML comments, JavaScript code, tags, and non-alphabetic characters are removed, as in [8].

Operation 2: Stemming is performed using the standard Porter stemmer [13].

Operation 3: Stopwords are removed by using the standard SMART stopword list [20].

Operation 4: Let W denote some predetermined window size. All query keywords in K_i are identified in Web page P. We only keep those words in P whose distances from a query keyword in K_i are no more than W words. All other words are discarded as noise.

Operations 1, 2, and 3 are standard operations in Web information retrieval. Operation 4 is specific for our purposes, as a Web page P in set S_i (i=1, 2) may contain a lot of irrelevant information. For example, P may contain several pieces of news, only one of which is related to entity E_i. If no content is dropped from P, too much noise may remain in P and make later analysis difficult. On the other hand, it is not desirable to drop too much content from P at the beginning. Otherwise useful information may be lost and relationships cannot be discovered.

Intuitively, Operation 4 attempts to strike a balance between noise reduction and omission of useful information. We assume that the most useful information is typically centered around query keywords and use windowing to obtain this information. Our experiments in Section 3 show that this windowing technique is important in practice and a good value for the window size W is usually between 25 and 35.

Our method treats all the Web pages in set S_i (i=1, 2) equally regardless of their original ranks provided by the Web search engine in Step 1. In general, the relevant Web pages that mention the relationships between entities E_1 and E_2 may be ranked low by the search engine. Our goal is to boost the ranks of these relevant Web pages so that they can appear early in the answers that are provided to the searcher. How to better utilize the original ranks provided by the search engine is left for future work.

2.5 Step 3: Computing Similarity Values

After pre-processing the two Web page sets S_1 and S_2, we use them to find the relationships between entities E_1 and E_2. For each pair of Web pages (P_1, P_2), where $P_1 \in S_1$ and $P_2 \in S_2$, we compute a similarity value that reflects the likelihood that (P_1, P_2) mentions some relationship between E_1 and E_2. Also, for each word t that appears in both P_1 and P_2, we compute a term weight that reflects the likelihood that t captures the relationship between E_1 and E_2.

Following the convention of information retrieval literature [4], we define *vocabulary* as the set of all the distinct words. A *term* is a word. Moreover, we define *connecting terms* as terms that capture the relationships between entities E_1 and E_2 [10]. When weights are properly assigned, terms with larger weights are more likely to be connecting terms.

We propose an enhanced version of the state-of-the-art Okapi formula [17, 18] to compute both term weights and the similarity values of Web page pairs. We first give a brief summary of Okapi. In Okapi, both documents and queries are represented as vectors. Each element of a vector is the weight of a term in the vocabulary. Terms that are important to a document are assigned large weights. Terms that do not appear in the document have weights zero. The relevance between a document D and a query Q is computed as the inner product of D's vector and Q's vector.

The intuition behind Okapi is that the more times a term t appears in a document D and the fewer times t appears in other documents (i.e., the less popular t is in other documents), the more important t is for D. Also, the effect that longer documents have more words needs to be compensated by normalizing for document lengths.

Consider a query Q and a document set S. For each term t in the vocabulary and a document $D \in S$, Okapi uses the following formulas:

(f1) term frequency (tf) weight

$$w_{tf} = (k_1 + 1)tf / \{k_1[(1-b) + b \times dl / avdl] + tf\},$$

(f2) inverse document frequency (idf) weight

$$w_{idf} = \ln[(N - df + 0.5)/(df + 0.5)],$$

(f3) query term frequency weight

$$w_{qtf} = (k_3 + 1)qtf / (k_3 + qtf),$$

(f4) term weight $w_t = w_{tf} \times w_{idf} \times w_{qtf}$,

(f5) $score_{D,Q} = \sum_{t \in D,Q} w_t$.

Here tf is t's frequency (i.e., number of occurrences) in D, qtf is t's frequency in Q, N is the total number of documents in S, df is the number of documents in S that contain t, dl is the length of D in bytes, and $avdl$ is the average length (in bytes) of all the documents in S. b, k_1, and k_3 are three predetermined constants. Typically, as suggested in [18], $b=0.75$, $1 \le k_1 \le 2$, and $1 \le k_3 \le 1000$.

For each document $D \in S$, Okapi defines its score (i.e., the degree of relevance for answering query Q) as in equation f5. This score is the sum of term weights of all the terms that appear in both D and Q.

For RQs, we deal with two Web page sets rather than one document set and one query. Thus, we modify Okapi to compute the similarity value between two Web pages, where either Web page comes from a different set. (Note that a query can be treated as a document [4]. A Web page is also a document.) Our idea is to replace (D, Q) with ($P_1 \in S_1$, $P_2 \in S_2$) and exploit the symmetry between the two Web page sets S_1 and S_2. At a high level, our method reuses equation f1, drops f3, and changes f2, f4, and f5 into f2', f4', and f5', respectively.

(f2') revised idf weight $w'_{idf} = \ln[(N + 0.5)/(df + 0.5)]$,

(f4') revised term weight

$$w'_t = w_{tf,1} \times w_{tf,2} \times \max(w'_{idf,1}, w'_{idf,2}),$$

(f5') $sim_{P_1,P_2} = \sum_{\substack{t \in P_1,P_2 \\ top\ C}} w'_t$.

The rationale of changing the idf weight from equation f2 to f2' is as follows. Okapi deals with one large document set while we have two small Web page sets: S_1 and S_2. If a term t appears in a large number of Web pages in S_1 and S_2 (e.g., $df > N/2$), Okapi computes a negative idf weight for t, which is not desirable for our purpose. Note that in a traditional large document set, this problem is not likely to occur. However, in our case, all the Web pages in the small set S_i (i=1, 2) are on the same topic E_i and thus some popular terms related to E_i may appear in a large portion of these Web pages. To avoid running into this negative value problem, we drop the term $-df$ and re-define the idf weight as in equation f2', which is similar to the traditional idf weight $w_{idf} = \ln(N/df)$ that is described in [4].

Although Okapi and our method use similar formulas to compute the tf weight and the idf weight, they use different ways to compute the global statistics: the number N of documents, the average document length $avdl$, and the document frequency df. In Okapi, these global statistics are computed on the entire document set. In our method, for either i (i=1, 2), we compute a set of global statistics (N_i, $avdl_i$, df_i) on the Web page set S_i. Consider a pair of Web pages (P_1, P_2), where $P_1 \in S_1$ and $P_2 \in S_2$. For each term t in the vocabulary and either P_i (i=1, 2), we use (N_i, $avdl_i$, df_i) to compute a tf weight $w_{tf,i}$ and an idf weight $w'_{idf,i}$. In this way, we can capture various characteristics of S_1 and S_2 that reflect different properties of the two entities E_1 and E_2.

Our way of computing the term weight (equation f4') is different from that in Okapi (equation f4). Each common term t that is shared by Web pages P_1 and P_2 is a potential connecting term. It carries some information about the likelihood that some relationship between entities E_1 and E_2 is mentioned in (P_1, P_2). We need to compute the term weight, or the contribution of t to the similarity value of (P_1, P_2). This term weight also reflects the likelihood that t is a real connecting term. Just like the intuition behind Okapi, the more times t appears in P_i (i=1, 2), the more important t is for (P_1, P_2). Hence, both $w_{tf,1}$ and $w_{tf,2}$ should appear in our term weighting formula, say by multiplying them together.

However, it is unfair to state that the fewer times t appears in the other documents in sets S_1 and S_2, the more important t is for (P_1, P_2). The reason is as follows. Recall that one intuition behind Okapi is that popular terms are unimportant terms. That is for the case of a single document set. In our case, there are two Web page sets: S_1 and S_2. If a term t is popular in one Web page set (e.g., S_1) but not in the other (e.g., S_2), t can be highly correlated with E_1 but not a generally popular term. In general, as long as t is unpopular in one of the two sets S_1 and S_2, t can be an important connecting term in the Web page pair (P_1, P_2). To take this into consideration, our term weighting formula uses $\max(w'_{idf,1}, w'_{idf,2})$ rather than $w'_{idf,1} \times w'_{idf,2}$, and the term weight is re-defined as in equation f4'. Note that S_1 and S_2 are symmetric in our case. This requires our term weighting formula to be symmetric with respect to 1 and 2. Equation f4' fulfills this requirement.

Now we find all the common terms that are shared by Web pages P_1 and P_2. These common terms are potential connecting terms. They are sorted in descending order of w'_t's, i.e., the estimated likelihoods that they are real connecting terms.

Finally, for each pair of Web pages (P_1, P_2), where $P_1 \in S_1$ and $P_2 \in S_2$, their similarity value is computed as in equation f5'. This similarity value is the sum of the contributions of the top C potential connecting terms. It is an approximation to the likelihood that some relationship between entities E_1 and E_2 is mentioned in (P_1, P_2). Note that this computation considers only the top C potential connecting terms rather than all the potential connecting terms. We propose this filtering technique to reduce noise in long Web pages. In general, those common terms with small weights w'_t are unlikely to be real connecting terms. A pair of long Web pages (P_{l1}, P_{l2}) are likely to share a large number of common terms (possibly with small weights w'_t), even if (P_{l1}, P_{l2}) does not mention any relationship between E_1 and E_2. If all the common terms are considered, due to the large number of such terms, (P_{l1}, P_{l2}) is likely to have a large similarity value. This is misleading. In contrast, in the case of a traditional document set S and a query Q, for each document $D \in S$, Q is always the same. Hence, this filtering technique is unnecessary in Okapi.

In practice, if C is too small, the computed similarity value cannot capture enough useful information. On the other hand, if C is too large, a lot of noise may be introduced into the computed similarity value. Our experiments in Section 3 show that a good value for C is usually between 20 and 30.

2.6 Step 4: Sorting Web page Pairs

After computing the similarity values, all the Web page pairs are sorted in descending order of their similarity values, i.e., the estimated likelihoods that they mention some relationships between entities E_1 and E_2. The top ten Web page pairs are returned to the searcher in the first result page. For each Web page pair (P_1, P_2), the common terms of P_1 and P_2 are sorted in descending order of their weights w_t', i.e., the estimated likelihoods that they are real connecting terms. The top 15 common terms (if there are so many common terms) are displayed as potential connecting terms.

As mentioned in Section 2.1, when the searcher views the entire Web page P_i (i=1, 2), both the query keywords in set K_i that are not in the stopword list and the 15 potential connecting terms are highlighted in P_i. In general, the relationship between E_1 and E_2 is likely to be mentioned somewhere close to those terms. This highlighting can facilitate the searcher to find the relationship. Also, not all the common terms of P_1 and P_2 are highlighted, as there may be too many such terms and highlighting all of them will get the searcher overwhelmed. The number 15 is an empirical number. We find that 15 is usually sufficient for highlighting useful potential connecting terms without overwhelming the searcher with too much information.

2.7 Discussions

The above descriptions show how to implement our method on top of a Web search engine. More efficiently, our method can be implemented inside a Web search engine if the search engine code is accessible. For example, Step 1 (obtaining Web pages) can be performed locally, as the Web search engine has a local copy of all the Web pages. Also, Operations 1, 2, and 3 of Step 2 (document pre-processing) can be done beforehand for all the Web pages. Then there is no need to repeat these operations for each individual RQ.

Currently, we only consider a few parameters. There are many additional parameters that could be included, such as Web page titles, weighting schemes for font sizes, and link information. Our goal is not to exhaustively search the entire parameter space. Rather, a reasonable set of parameters are chosen to demonstrate that our general methodology is promising and can achieve good performance in many cases. How to use additional parameters to improve the effectiveness of our method is left for future work.

Our current method works for answering RQs on the Web. It may also work for answering RQs on traditional document sets. A thorough investigation of this issue is left for future work. In Section 3, to show that our general methodology is promising, we provide an example of applying our techniques to the problem of finding relationships in two document sets, where the concept of RQ is generalized.

In summary, our approach has the following advantages (see Section 3 for details):

(1) It can find multiple relationships between two entities simultaneously with low overhead (typically less than one second). For example, each returned top Web page pair may mention a different relationship.

(2) It can find important relationships even in the presence of a lot of "noise."

(3) The quality of search results is insensitive to the parameter values used in our method within reasonable ranges.

(4) It can find different kinds of relationships, such as commonalities, differences, contrasts, direct connections, and indirect connections.

(5) It can be implemented either inside or on top of a Web search engine. In the latter case, it can be implemented on either the client side or the server side.

(6) Its interface is user friendly. Searchers can keep using the familiar keyword query interface of Web search engines.

(7) It can be the basis for extracting exact answers to RQs, regardless of whether the exact answers are short answers or long answers. Here the short answer means that the exact answer is composed of one or a few sentences in a Web page pair. The long answer [3] means that the exact answer is composed of one or a few paragraphs in a Web page pair, or even the entire Web page pair (e.g., detailed news report).

3. EXPERIMENTAL RESULTS

We have implemented a prototype of the proposed techniques on top of the Google search engine [7]. Currently, there is no standard benchmark for answering RQs on the Web. The RQs used in AQUAINT [2] and TREC [15] are for traditional document sets. We have conducted extensive experiments with those RQs. Since the Web contains much more information than traditional document sets and existing Web search engines excel in keyword matching, the answers for most of those RQs can be easily found on the Web by typing both entities' keywords in a Web search engine.

Based on the examples in AQUAINT and TREC [2, 15] as well as experience that we learned through interviews with Web searchers, we built our own test set of RQs for experimentation – a total of 30 examples that are classified into various scenarios. (Note that only 25 RQs were used for the relationship task in TREC 2005.) The detailed results of seven examples from five representative scenarios and the average results for all the 30 examples are provided. According to our prototyping experience, detailed results are necessary to help the reader get clear insight into many important issues (e.g., how queries and results look like, how meaningful the results are, when and why our techniques succeed or fail). None of the 30 examples can be well handled by existing Web search engines (including Google). All our experiments were performed between Sep. and Oct. 2006. According to our tests, the precise query form (e.g., whether quotation mark is used) has minor impact on the found relationships. In this section, we only present the results for the most basic query form (without quotation mark). As the Web and Google search results keep changing over time, our results may have minor changes while the found relationships will remain roughly the same.

In this section, $P_{i,j}$ denotes the jth Web page that is retrieved from the Google Web search engine for entity E_i (i=1, 2). Each Web page pair is represented in the format in Figure 4. (Example 7 is an exception, where there is no concept of URL.) The default parameter values used in our method are as follows: M_1=M_2=50 (the number of top Web pages retrieved from a Web search engine for an entity), W=30 (the window size used in document pre-processing), k_1=1.2 (the parameter in the tf weight computation formula), and C=20 (the number of top potential connecting terms considered in computing the similarity value of a Web page pair). The effect of various parameter values on the answer quality is discussed in Section 3.2.

Web page 1	Web page 2
URL 1	URL 2
Potential Connecting Terms	

Figure 4. Web page pair format.

3.1 Examples

Scenario I: Relationship between People

Example 1 (Nobel Example)

This is the example mentioned in the introduction. Glenn Klausman, the lawyer, would like to find out the relationship between himself and John Robert Schrieffer, the Nobel Prize laureate. In this experiment, the keywords for entity E_1 are *Glenn Klausman*. The keywords for entity E_2 are *John Robert Schrieffer*.

Figure 3 shows the returned first Web page pair $(P_{1,1}, P_{2,28})$. Recall that $P_{2,28}$ is the 28th Web page retrieved from Google for entity E_2. The top few potential connecting terms include injury and court. They provide a good hint for the relationship between Glenn and John. From this Web page pair, Glenn can easily find out that John once ran into a car accident.

Note that Web search engines make mistakes in certain cases. For example, not all top 50 Web pages that Google retrieved for entity E_1 are related to attorney Klausman (such as www.library.uiuc.edu/rex/erefs/bronzetablets/1960s.htm). Our method can automatically filter out such noise and find the right information. Moreover, in a relevant top Web page pair that contains the desired relationship between the two entities, one or both Web pages may be originally ranked low (e.g., 28th in this example) by the Web search engine. Our method is able to boost the rank of relevant Web pages.

Example 2 (Lomet Example)

Arthur Ciccolo is the head of search technology at IBM Thomas J. Watson Research Center. David Lomet is the manager of the database group at Microsoft Research. Suppose Arthur will attend a conference and he notices that David will attend the same conference. Assume that Arthur does not know David and would like to chat with him. To get prepared, Arthur does some background search on the Web, attempting to find some relationship between him and David. In this experiment, the keywords for entity E_1 are *Arthur Ciccolo*. The keywords for entity E_2 are *David Lomet*.

Table 1 shows the returned first Web page pair $(P_{1,48}, P_{2,5})$. The tenth, eleventh, and twelfth potential connecting terms are related to IBM T.J. Watson Research Center. From this Web page pair, Arthur can easily find out the relationship between him and David – they both have worked at IBM T.J. Watson Research Center.

Table 1. Returned first Web page pair of Example 2.

… **Arthur** C. **Ciccolo** IBM Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (ciccolo@us.ibm.com). Mr. **Ciccolo** is a Department Group Manager in the Research Division of IBM and co-leader of the IBM Institute for Search and Text Analysis ...	… **DAVID LOMET** has been a senior researcher and manager of the Database Group at Microsoft Research, Redmond, Washington since 1995. ... Earlier, he was a researcher at the IBM Thomas J. Watson Research Center in Yorktown ...
www.research.ibm.com/journal/sj/433/brodeaut.html	www.ccs.neu.edu/colloquium/lomet.html
award, IEEE, paper, chair, committee, work, serve, patent, speaker, Watson, IBM, Thomas …	

Now suppose we replace Arthur Ciccolo with Jennifer Chu-Carroll, a Research Staff Member at IBM T.J. Watson Research Center. That is, the keywords for entity E_1 become *Jennifer Chu-Carroll*. In this case, the top four Web page pairs are mainly about paper collections and conferences that are irrelevant to the relationship between Jennifer and David. Only from the fifth Web page pair (www.naacl.org/elections/jc-2005.html, www.ccs.neu.edu/colloquium/lomet.html), Jennifer can find out that both she and David have worked at IBM T.J. Watson Research Center.

Many Web pages that Google returns for Jennifer are noisy and contain little information about her (e.g., conference PC name list). Consequently, the case of Jennifer is more difficult than the case of Arthur and the found relevant Web page pair is ranked lower.

Scenario II: Relationship between Places

Example 3 (Yorktown Example)

Suppose Mary gets two job offers, one at Yorktown Heights, NY, and another at Shorewood Hills, WI. To decide which job offer to accept, Mary would like to compare these two places and see which place she likes more. In this experiment, the keywords for entity E_1 are *Yorktown Heights*. The keywords for entity E_2 are *Shorewood Hills*.

Table 2 shows the first Web page pair $(P_{1,31}, P_{2,21})$, which provides some useful comparison (population, latitude, and businesses) between Yorktown Heights and Shorewood Hills.

Table 2. Returned first Web page pair of Example 3.

… Facts & Statistics Place Name: **Yorktown Heights** … Population: 7,690 (1990) Location: Westchester County, New York (NY), United States … Latitude: 41°16'N Longitude: 73°46'W ... IBM's T. J. Watson Research Center is located here …	… Facts & Statistics Place Name: **Shorewood Hills** … Population : 1,680 (1990) Location: Dane County, Wisconsin (WI), United States … Latitude: 43°04'N Longitude: 89°27'W ... At W end of Univ. of Wis. Campus …
reference.allrefer.com/gazetteer/Y/Y01318-yorktown-heights.html	reference.allrefer.com/gazetteer/S/S10841-shorewood-hills.html
Columbia, press, gazetteer, university, America, related, symbol, seat, …	

Table 3 shows the second Web page pair $(P_{1,46}, P_{2,19})$, which gives a comparison of the weather condition between Yorktown Heights and Shorewood Hills. Table 4 shows the other useful Web page pairs in the top ten Web page pairs.

Table 3. Returned second Web page pair of Example 3.

Yorktown Heights, New York (10598) Weather ... Today...Mostly sunny. Highs in the mid 80s. Northwest winds 5 to 10 mph ... Saturday...Mostly sunny. Highs in the lower 80s. Northwest winds 5 to 10 mph ...	**Shorewood Hills**, WI Weather Forecast … FRIDAY - Mostly sunny. Highs in the upper 70s. Northwest winds 5 to 15 mph … SATURDAY - Partly cloudy. Highs in the upper 70s. North winds up to 5 mph ...
weather.allrefer.com/new-york/yorktown-heights.html	www.city-data.com/forecast/w-Shorewood-Hills-Wisconsin.html
clear, wind, forecast, EDT, SEP, weather, mph, partly, cloudy, night, …	

Table 4. Returned other useful Web page pairs of Example 3.

Web page pair	URLs of the Web page pair	relationship
third $(P_{1,1}, P_{2,9})$	(www.city-data.com/city/Yorktown-Heights-New-York.html, www.city-data.com/city/Shorewood-Hills-Wisconsin.html)	detailed comparison
eighth $(P_{1,13}, P_{2,24})$	(www.sublet.com/area_rentals/NewYork/YorktownHeights_Rentals.asp, www.rentspeed.com/cities/WI_Shorewood+Hills_Wisconsin.aspx)	apartment rental
tenth $(P_{1,8}, P_{2,28})$	(www.homegain.com/local_real_estate/NY/yorktown_heights.html, www.realestate.com/cityengine/WI/Shorewood%20Hills.html)	real estate

In this example, each of the above mentioned five Web page pairs identifies a different relationship or comparison between Yorktown Heights and Shorewood Hills. That is, our method can find multiple relationships between two entities simultaneously. All the information is useful to Mary.

Example 4 (Hartlepool Example)

Hartlepool is a North Sea port in the United Kingdom. The Three Gorges region is a scenic area along the Yangtze River in China. Suppose Philip is an infrastructure protection analyst. He noticed that these two places were mentioned frequently in separate interesting reports, and would like to find out whether there is some connection between these two places**.** In this experiment, the keyword for entity E_1 is *Hartlepool*. The keywords for entity E_2 are *Three Gorges*.

Table 5. Returned fourth Web page pair of Example 4.

… A nuclear power station of the advanced gas-cooled reactor (AGR) type was opened near **Hartlepool** in the 1980s and is scheduled for decommissioning by 2014 ...	… One year ago, the 660-kilometre-long reservoir of the **Three Gorges** Project, the world's largest hydropower station, was successfully filled with water ...
en.wikipedia.org/wiki/Hartlepool	www.chinadaily.com.cn/english/doc/2004-07/15/content_348413.htm
village, resident, population, county, total, sport, project, schedule, …	

The top three Web page pairs are not very useful. Table 5 shows the fourth Web page pair $(P_{1,17}, P_{2,49})$. Note that both $P_{1,17}$ and $P_{2,49}$ are originally ranked low by Google. Although the relevant connecting term, station, is ranked low in all the potential connecting terms, Philip can find out from this Web page pair that there is a nuclear power station in Hartlepool and the world's largest hydropower station (dam) is in Three Gorges. Namely both places have important objects (nuclear power station and dam), the destruction of which can lead to a disaster.

Note that Example 3 (Yorktown Example) and Example 4 (Hartlepool Example) are rather different. The two places in Example 3 have only loose connections. In contrast, the two places in Example 4 have much stronger connections. Our method is self-adaptive and can find desired relationships for both examples.

Scenario III: Relationship between Companies
Example 5 (Bank Example)

Union Bank of Switzerland is a major bank in Switzerland. It is known that some criminals deposit their money in Swiss banks. St. Petersburg Real Estate Holding Co. is a Germany-based company that buys real estates in St. Petersburg, Russia. Suppose Philip is a financial auditor. Based on some financial transaction evidence, he suspects that there are some connections between St. Petersburg Real Estate Holding Co. and Union Bank of Switzerland. Philip would like to find out the connections. In this experiment, the keywords for entity E_1 are *St. Petersburg Real Estate Holding Co.* The keywords for Entity E_2 are *Union Bank of Switzerland*. (Note that the bank name and the company name are only used for illustration purposes rather than implying a fact.)

Using the keywords of *Union Bank of Switzerland*, the top 50 Web pages returned from Google are all about the bright side of Union Bank of Switzerland, e.g., bank merge information. Since Philip is only interested in "dark-side" relationships, he changes the keywords for entity E_2 to *Union Bank of Switzerland scandal* by adding the word "scandal." Now Philip can get access to the desired information. (Note that existing Web search engines cannot use the keywords for both entities E_1 and E_2 to find desired relationships between E_1 and E_2, irrespective of how those keywords are chosen.)

Table 6. Returned first Web page pair of Example 5.

Caught in the center of a Germany-wide money-laundering investigation is a **St. Petersburg real estate** company ... But a German prosecutor told Germany's Der Spiegel magazine in Monday's edition that the raids were part of a two-year investigation into SPAG, or the **St. Petersburg Real Estate Holding Co.** ... SPAG was singled out in the German foreign intelligence investigation as a company suspected of laundering funds for Russian criminal gangs and Colombian drug lords …	… The **bank** also had friends in high places in the U.S. … Over the years, BCCI was involved with: • Drug cartels. As early as 1985, the U.S. Drug Enforcement Administration (DEA) and the IRS found that BCCI was involved in laundering heroin money, with numerous branches in Colombia to handle accounts for the drug cartels ... Arkansas investment banker Jackson Stephens in 1987 worked out Harken's debts by getting $25 million financing from **Union Bank of Switzerland** (UBS), a partner with BCCI ...
www.cdi.org/russia/johnson/7187-11.cfm	www.alternet.org/election04/20268/
launder, crime, prosecutor, investigation, traffick, criminal, money, arm, indictment, intelligence, fine, drug …	

Table 6 shows the first Web page pair $(P_{1,15}, P_{2,45})$. The top two potential connecting terms are related to money laundering crime. From $P_{1,15}$, Philip finds out that St. Petersburg Real Estate Holding Co. has been suspected of laundering money for Columbia drug lords. From $P_{2,45}$, Philip finds out that Union Bank of Switzerland is a partner with BCCI (Bank of Credit and Commerce International), and BCCI is involved in money laundering for Columbia drug cartels. Hence, there is an indirect relationship between St. Petersburg Real Estate Holding Co. and Union Bank of Switzerland – both of them are (suspected of being) directly or indirectly connected to money laundering for Columbia drug cartels.

Scenario IV: Relationship between Institutes
Example 6 (CMU Example)

Table 7. Returned first Web page pair of Example 7.

Microsoft Research Redmond, Washington ... **Research** Units: Algorithms and Theory Human-Computer Interaction Machine Learning, Adaptation and Intelligence Multimedia and Graphics Search, Retreival and Knowledge Management Security and Cryptography Social Computing Software Development Systems, Architectures, Mobility, and Networking ...	**Carnegie Mellon University** (CMU) Pittsburgh, PA ... Research Units: School of **Computer Science** Information Networking Institute Robotics Institute Department of Electrical & **Computer** Engineering (ECE) … CyLab Data Storage Systems Center The Sage Visualization Group Human-**Computer** Interaction Institute Advanced Multimedia Processing Lab Natural Language Processing SPIRAL ...
www.trnmag.com/Directory/Query_Results/Corporate/Microsoft_Research_Computing.html	www.trnmag.com/Directory/Query_Results/University/Carnegie_Mellon_University_Computing.html
trn, category, deeper, LLC, aspect, reserve, cover, …	

Suppose Anton graduated from the Computer Science Department of Carnegie Mellon University (CMU) and is currently a researcher at Microsoft Research (MSR). He will go back to CMU to recruit new employees for MSR. On that trip, Anton will meet with a few faculty members at CMU. To get prepared, Anton would like to find out the common interests (e.g., research areas) between MSR and CMU. In this experiment, the keywords for entity E_1 are *Microsoft Research*. The keywords for entity E_2 are *Carnegie Mellon University computer science*.

Table 7 shows the first Web page pair $(P_{1,39}, P_{2,11})$, which provides a detailed list of computer science research areas in MSR or CMU. Links to the corresponding research units are provided, and some of the CMU research units are not easily accessible from the homepage of School of Computer Science at CMU.

Scenario V: Relationship between Document Sets
Example 7 (Paper Example)

The following example shows that our techniques are not limited to answering RQs on the Web. Here we demonstrate the generality of our techniques by applying them to traditional document sets. No entity exists in this example. Rather, we are interested in finding relationships in two document sets and the concept of RQ is generalized.

Suppose Cathy is a manager at a research lab. She recently becomes interested in the database area because of the nature of some on-going projects in her team. Cathy would like to see whether there is any collaboration opportunity between her team and the database research community. However, neither Cathy nor people in her team are familiar with the database area. To make this up, Cathy plans to send the best matching people in her team to attend the SIGMOD'05 conference. Cathy has two document sets: S_1 and S_2. S_1 is the collection of papers written by people in her team. S_2 is the collection of 84 papers published in SIGMOD'05.

By matching documents in S_1 with documents in S_2, our techniques can help Cathy find the best matching people in her group. The concrete method is as follows. Without loss of generality, we assume that Cathy's group works on operating system and the document set S_1 is the collection of 49 papers published in OSDI'04 (Symposium on Operating Systems Design and Implementation) and SOSP'03 (ACM Symposium on Operating Systems). On one hand, the paper title alone does not contain enough information, and the entire paper may introduce too much noise. On the other hand, the title and the abstract often give a good summary of the content in the paper. Hence, for each paper in S_1 or S_2, instead of using the entire paper, only the title and the abstract are used. All the papers in S_1 and S_2 are used. Since we are not retrieving Web pages from the Web, there is no need to perform Step 1 (obtaining Web pages) and Operation 1 in Step 2 (document pre-processing). Also, Operation 4 in Step 2 is omitted, as no entity exists in this case. Everything else remains the same, as described in Section 2.

Table 8 shows the first document pair, which matches an overlay network paper with a sensor network paper. The identified potential connecting terms show that both papers are related to data streaming and networking. Actually, the multicast approach (one point to multiple points) used in overlay network can be regarded as the reverse procedure of the aggregation approach (multiple points to one point) used in sensor network. Cathy can send the authors of the overlay network paper to SIGMOD'05, with a particular interest in data streaming in sensor networks.

Table 8. Returned first document pair of Example 7.

Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh Abstract In recent years, overlay networks have become an effective alternative to IP multicast for efficient point to multipoint communication across the Internet. Typically, nodes self-organize with the goal of forming an efficient overlay tree … In this paper, we target high-bandwidth data distribution from a single source to a large number of receivers. Applications include large-file transfers and real-time multimedia streaming ...	Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams Abstract Existing energy-efficient approaches to in-network aggregation in sensor networks can be classified into two categories, tree-based and multi-path-based, with each having unique strengths and weaknesses. In this paper, we introduce Tributary-Delta, a novel approach that combines the advantages of the tree and multi-path approaches by running them simultaneously in different regions of the network ...
tree, network, algorithm, item, rate, factor, efficient, simultaneously, stream, data, …	

The second, third, fourth, and sixth document pairs match the overlay network paper with the following four data stream papers, respectively: RPJ: Producing Fast Join Results on Streams through Rate-based Optimization, Sampling Algorithms in a Stream Operator, Conceptual Partitioning: an Efficient Method for Continuous Nearest Neighbor Monitoring, and Holistic Aggregates in a Networked World: Distributed Tracking of Approximate Quantiles. The authors of the overlay network paper may also be interested in these four data stream papers.

The fifth document pair matches a network monitoring paper (Ksniffer: Determining the Remote Client Perceived Response Time from Live Packet Streams) with a data stream paper (Sampling Algorithms in a Stream Operator). Both papers are related to data streaming. If budget permits, Cathy may also send the authors of the network monitoring paper to SIGMOD'05, with a particular interest in data streams.

For all the 30 examples we used in our experiments, desired relationships can be found in the returned top ten Web page pairs. For 24 of these 30 examples, desired relationships can be found in the returned top three Web page pairs. The detailed results of the other 23 examples are similar to those of the seven examples presented above and omitted due to space constraints. The average results for all the 30 examples can be found in Section 3.2.

Overhead of Our Method

Recall that as mentioned in Section 2.7, our method can be implemented more efficiently inside a Web search engine. In this case, Operations 1, 2, and 3 of Step 2 (document pre-processing) can be done beforehand for all the Web pages. Step 1 (obtaining Web pages) is necessary irrespective of which method is used to answer RQs. Hence, the additional overhead of our method is Operation 4 of Step 2 (windowing), Step 3 (computing similarity values), and Step 4 (sorting Web page pairs). According to our measurements on an IBM ThinkPad T40 PC with one 1.6GHz processor, 1GB main memory, one 75GB disk, and running the Microsoft Windows XP operating system, this additional overhead is always less than one second in all the examples.

3.2 Sensitivity Analysis of Parameter Values

There are several important parameters in our method. In this section, we evaluate the impact of parameter values on the quality of answers by a set of experiments. In each experiment, we varied the value of one parameter while keeping the other parameters unchanged. (The case of M_1 and M_2 is an exception, where the values of two parameters are changed simultaneously.)

For each RQ, a *score* is calculated to evaluate the quality of the returned Web page pairs. This score is defined as the sum of reciprocal ranks of relevant Web page pairs in the returned top ten Web page pairs [16], where relevant Web page pairs contain desired relationships between the two entities and are manually identified. For example, if in the returned top ten Web page pairs, the first, second, and eighth Web page pairs are relevant ones, the score would be $1+1/2+1/8=1.625$. As mentioned in [16], this score is a reasonable measure of ranking method performance, as it favors relevant Web page pairs that are ranked higher while also giving appropriate weights to lower ranked relevant Web page pairs. Also, this score considers the possibility that multiple relationships exist between the two entities and thus multiple relevant Web page pairs may be found simultaneously.

Recall that we have 30 examples in total. The average score for the RQs in these 30 examples is a single-value indicator of the performance of the ranking method. In each experiment, the average score is reported to show the sensitivity of our method to the changed parameter.

M_1 and M_2 (Number of Retrieved Top Web pages)

The first experiment concerns M_1 and M_2, the numbers of top Web pages retrieved from a Web search engine for both entities (Step 1). The default values of M_1 and M_2 are 50. We varied M_1=M_2 from 25 to 100. Figure 5 shows the impact of M_1=M_2 on the average score. (Note: to make figures in Sections 3.2 and 3.3 more readable, the y-axis does not always start from zero.) In general, when M_1 and M_2 are too small, there may not be enough Web pages to discover useful information. When M_1 and M_2 become larger, more Web pages are retrieved from the Web search engine. This may lead to the discovery of more useful information. On the other hand, this also has the danger of not being able to discover any useful information in the returned top few Web page pairs, as more Web pages introduce more noise and thus make the Web page pair ranking process more difficult. Choosing M_1=M_2=*50* is usually sufficient for discovering important relationships between two entities without making the ranking process too difficult.

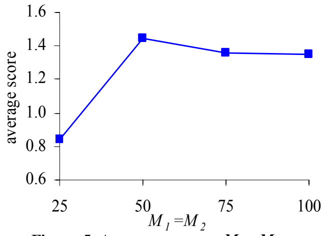


Figure 5. Average score vs. M_1=M_2.

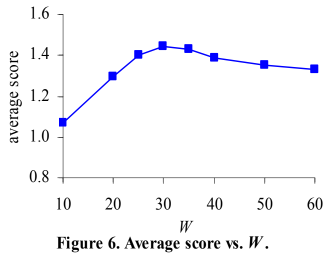


Figure 6. Average score vs. *W*.

***W* (Window Size)**

The second experiment concerns *W*, the window size used in document pre-processing (Operation 4 of Step 2). The default value of *W* is 30. We varied *W* from 10 to 60. Figure 6 shows the impact of *W* on the average score. When *W=25* or *W=35*, the answers are basically the same as that when *W=30*. In general, when *W* is too small, useful information in the Web pages may get lost. When *W* is too large, a lot of noise may remain in the Web pages. The safe range for *W* is between 25 and 35. If *W* is outside of this safe range, the answer quality will degrade.

k_1 (Parameter in the tf Weight Computation Formula)

The third experiment concerns k_1, the parameter in the tf weight computation formula (Step 3). The default value of k_1 is 1.2. We varied k_1 from 0.5 to 3. Figure 7 shows the impact of k_1 on the average score. When k_1=*1* or k_1=*1.5*, the answers are basically the same as that when k_1=*1.2*. In other words, the safe range of k_1 is between 1 and 1.5, which is smaller than the range of [1, 2] that was reported in [18].

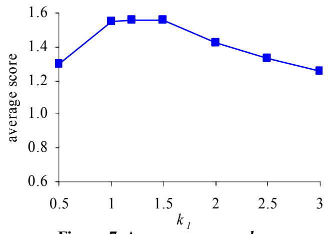


Figure 7. Average score vs. k_1.

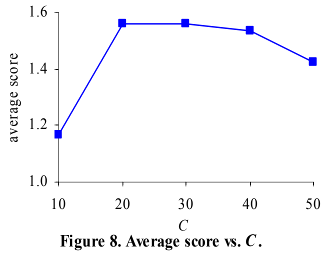


Figure 8. Average score vs. *C*.

***C* (Number of Top Potential Connecting Terms)**

The fourth experiment concerns *C*, the number of top potential connecting terms considered in computing the similarity value of a Web page pair (Step 3). The default value of *C* is 20. We varied *C* from 10 to 50. Figure 8 shows the impact of *C* on the average score. When *C=30*, the answers are basically the same as that when *C=20*. In general, when *C* is too small, not enough useful information is captured in the computation process of similarity values of Web page pairs. When *C* is too large, a lot of noise may be introduced into that computation process. The safe range for *C* is between 20 and 30.

In summary, each of the parameters has a not-very-small safe range. That is, the answer quality is insensitive to parameter changes. However, if a parameter value is outside its safe range, the answer quality may degrade.

3.3 Influence of Individual Techniques

Our method uses the following key techniques:

(1) **Technique 1**: Use windowing in document pre-processing (Operation 4 in Step 2).

(2) **Technique 2**: Use $\max(w'_{idf,1}, w'_{idf,2})$ in the term weighting formula (Step 3).

(3) **Technique 3**: Only consider the top *C* potential connecting terms in computing the similarity value of a Web page pair (Step 3).

(4) **Technique 4**: For either *i* (*i*=1, 2), compute a set of global statistics (N_i, $avdl_i$, df_i) on the Web page set S_i (Step 3).

In this section, we discuss the influence of individual techniques on the answer quality. We performed a set of experiments. In each experiment, we dropped a single technique while keeping the other techniques unchanged. When Technique 1 is not used, all the words are kept in Web pages. When Technique 2 is not used, $w'_{idf,1} \times w'_{idf,2}$ is used in the term weighting formula (Step 3). When Technique 3 is not used, all the potential connecting terms are considered in computing the similarity value of a Web page pair. When Technique 4 is not used, we compute only one set of global statistics (*N*, *avdl*, *df*) on $S_1 \cup S_2$.

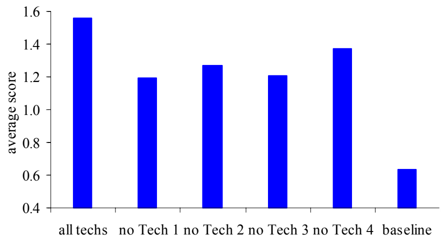


Figure 9. Average score vs. used techniques.

Figure 9 shows the impact of the used techniques on the average score. In this figure, "tech" stands for technique. "No Tech *i*" (*i*=1, 2, 3, 4) represents the case that Technique *i* is not used. Baseline represents the case that none of the four techniques is used. All techniques in our method are necessary. If any of them is not used, the quality of the answers will degrade. Techniques 1 and 3 are more important than Techniques 2 and 4 in the sense that they have a larger impact on the answer quality. Also, the performance of our method is much better than that of the baseline.

4. RELATED WORK

To the best of our knowledge, [9, 10, 24] are the only published work on answering general RQs. They focus on traditional document sets rather than the Web. The method proposed in [10] has some limitations when working with Web pages on the Web. For example, that method first forms a query *Q* that contains both entities' keywords and uses *Q* to retrieve 25 documents from a search engine. In the case of a Web search engine, this often leads to the situation that either no document is returned or all returned documents are related to a single entity. Moreover, that method cannot discover useful but non-obvious information in the documents, as in the document preprocessing step, that method only keeps the top sentences that are most similar to *Q*. The same limitations also exist for the methods proposed in [9, 24]. In bioinformatics, [19, 21] used domain-specific knowledge and the MEDLINE biomedical literature database to find relationships between two biomedical entities. However, those methods proposed in [19, 21] do not work for general RQs.

[5, 8] proposed a set of techniques for finding Web pages that are similar to a given Web page. Essentially, this is to find those Web pages that are on the same topic as the given Web page. In our case, we need to retrieve Web pages that are on different topics (i.e., related to different entities) and make connections between these Web pages. Therefore, those techniques proposed in [5, 8] cannot be used directly for our purpose.

[8] uses windowing around anchor texts to find Web pages that are similar to a given Web page. In our method, windowing around query keywords is used.

[23] proposed a set of techniques for finding terms that are correlated to one or more query terms. However, those found terms may not be strongly connected to the entire query. In our case, we need to find connecting terms that are strongly connected to both entities in the RQ. Also, just having those terms is far from being able to answer RQs.

Our work provides Web page pairs and potential connection terms as hints to the searcher. The searcher needs to further analyze these hints to find exact answers to RQs, while such an analysis is often easy for human beings. In contrast, in question answering [11], exact answers to queries are usually provided to the searcher directly. Since artificial intelligence is generally a hard problem, no satisfying question answering techniques currently exist for RQs.

In the database literature, [6] surveyed SQL-style query languages for the Web. However, none of these languages supports RQs. Also, [22] proposed extracting database relations from the Web, which are different from the relationships discussed in this paper.

5. CONCLUSION

We believe that we are among the first to study the problem of answering relationship queries on the Web. We proposed a method that matches top Web pages retrieved for individual entities and automatically identifies the connecting terms. To effectively filter out the large amount of noise in the Web pages without losing much useful information, we do windowing around query keywords, compute term weights based on the characteristics of the two Web page sets, and only use the top potential connecting terms to compute the similarity values of Web page pairs. Our experiments with a prototype implementation on top of the Google search engine show that our method is often effective at finding important relationships in a noisy environment with low overhead. The quality of search results is insensitive to parameter changes. Also, our method has a friendly user interface and can facilitate a wide range of searchers to explore the Web more efficiently.

6. ACKNOWLEDGEMENTS

We would like to thank Haiyan Chen, Tong Chen, Jeffrey F. Naughton, and John M. Prager for helpful discussions.

7. REFERENCES

[1] http://www.google.com/apis (Google APIs), 2005.
[2] http://www.itl.nist.gov/iaui/894.02/projects/aquaint (AQUAINT), 2005.
[3] S. Blair-Goldensohn, K. McKeown, and A.H. Schlaikjer. Answering Definitional Questions: a Hybrid Approach. New Directions in Question Answering 2004: 47-58.
[4] R.A. Baeza-Yates, B.A. Ribeiro-Neto. Modern Information Retrieval. ACM Press/Addison-Wesley, 1999.
[5] J. Dean, M.R. Henzinger. Finding Related Pages in the World Wide Web. Computer Networks 31(11-16): 1467-1479, 1999.
[6] D. Florescu, A.Y. Levy, and A.O. Mendelzon. Database Techniques for the World-Wide Web: a Survey. SIGMOD Record 27(3): 59-74, 1998.
[7] http://www.google.com (Google), 2005.
[8] T.H. Haveliwala, A. Gionis, and D. Klein et al. Evaluating Strategies for Similarity Search on the Web. WWW 2002: 432-442.
[9] S.M. Harabagiu, V.F. Lacatusu, and A. Hickl. Answering Complex Questions with Random Walk Models. SIGIR 2006: 220-227.
[10] D. Mahler. Holistic Query Expansion Using Graphical Models. New Directions in Question Answering 2004: 203-214.
[11] M.T. Maybury. New Directions in Question Answering. AAAI Press, 2004
[12] T.M. Mitchell. Machine Learning. McGraw Hill, 1997.
[13] M.F. Porter. An Algorithm for Suffix Stripping. Program 14(3): 130-137, 1980.
[14] J. Prange. Making the Case for Advanced Question Answering. Keynote Speech to Pragmatics of Question Answering Workshop at HLT/NAACL 2004.
[15] http://trec.nist.gov/data/qa/t2005_qadata.html (QA Collections of TREC), 2005.
[16] D.R. Radev, K.Libner, and W. Fan. Getting Answers to Natural Language Questions on the Web. JASIST 53(5): 359-364, 2002.
[17] S.E. Robertson, S. Walker, and M. Hancock-Beaulieu. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive. TREC 1998: 199-210.
[18] A. Singhal. Modern Information Retrieval: A Brief Overview. IEEE Data Eng. Bull. 24(4): 35-43, 2001.
[19] N.R. Smalheiser. The Arrowsmith Project: 2005 Status Report. Discovery Science 2005: 26-43.
[20] SMART Stopword List. http://www.lextek.com/manuals/onix/stopwords2.html, 2006.
[21] P. Srinivasan. Text Mining: Generating Hypotheses from MEDLINE. JASIST 55(5): 396-413, 2004.
[22] N. Sundaresan, J. Yi. Mining the Web for Relations. Computer Networks 33(1-6): 699-711, 2000.
[23] P. Tan, V. Kumar, and J. Srivastava. Indirect Association: Mining Higher Order Dependencies in Data. PKDD 2000: 632-637.
[24] http://trec.nist.gov/pubs/trec14/t14_proceedings.html. TREC 2005 Proceedings (relationship task in the QA track).