Journal of Convergence Information Technology
Volume 5, Number 6, August 2010

CT-Rank: A Time-aware Ranking Algorithm for Web Search

Peiquan Jin [*1], Xiaowen Li [1], Hong Chen [1], Lihua Yue [1]
[1] *School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230027, China, jpq@ustc.edu.cn*
doi: 10.4156/jcit.vol5.issue6.10

Abstract

Time plays important roles in Web search, because most Web pages contain time information and a lot of Web queries are time-related. However, traditional search engines such as Google have little consideration on the time information in Web pages. In particular, they do not take into account the time information of Web pages when ranking searching results. In this paper, we present a new time-aware ranking algorithm for Web search, which is called CT-Rank (Content-Time-based Ranking). The algorithm uses three factors of a Web page, namely the Pagerank value, the title ranking score, and the time-constrained keyword ranking score, to sort search results, and we develop a two-stage algorithm to realize the time-based ranking. We conduct a comprehensive experiment on 6,500 Web pages which is manually collected through Google, and compare the performance of CT-Rank with other four competitor algorithms including Pagerank, vector space model based ranking, update time based ranking, and Google's ranking algorithm. The experimental result shows that CT-Rank has the best performance under different temporal textual queries.

Keywords: *Ranking algorithm; Time information; Web search; Web page; Tf-idf*

1. Introduction

Nowadays, Web search engines such as Google and MSN have become necessary tools in people's daily life. Generally, the effectiveness of those search engines is mostly determined by some ranking algorithm, e.g., the *Pagerank* algorithm [1, 2] and HITS [1]. Unfortunately, traditional ranking algorithms are based on link analysis and keyword analysis [3], and are hard to satisfy the querying needs of users. For example, a user needs to search "Nike discount information in Geneva in the future week". Such queries are very difficult to be specified and evaluated in Google, MSN and other similar search engines. The main reason is that traditional search engines have not considered much on users' time-aware searching needs. In other words, time plays a central role in any information space, and it has been studied in other areas like information extraction, question-answering, and summarization [4]. Most Web pages contain time information. For example, Web pages may be updated frequently, so we have the last update time for each Web page. A News page usually report some events related with specific time instant or period. As a result, most Web pages contain *update time* and *content time* [5]. The update time refers to the last update time instant of Web page, while the content time is the time information reported in the content of Web page.

Therefore, if we enhance traditional ranking algorithms with time information in Web pages, it is expected that the search results will not only meet the keyword querying needs but also be constrained in some time range. As an extension to existing ranking techniques, time can be valuable for placing search results and improving the effectiveness of Web search engines.

In this paper, we present a new time-aware ranking algorithm for Web search, which is called CT-Rank (Content-Time-based Ranking). The algorithm combines keywords, update time and content time of Web page into the ranking procedure, and can improve the effectiveness of Web search engines. The contributions of the paper can be summarized as follows:

(1) We present a time-aware ranking algorithm that considers not only text relevance but also time relevance of Web pages. By introducing update time and content time of Web page into the ranking algorithm, our algorithm makes the ranking results more reasonable and improves the effective of searching results.

(2) We develop a two-staged strategy to implement the CT-Rank algorithm, which refers to an offline stage extracting and building <*keyword*, *time*, *score*> pairs for Web page and an online stage computing the final ranking scores based on *pagerank* value, title contribution, and time-constrained *tf-idf* value for each keyword.

(3) We conduct a comprehensive experiment on 6,500 Web pages which is manually collected through Google, and compare the performance of CT-Rank with other four competitor algorithms including *pagerank*, vector-model-based ranking, update-time-based ranking, and Google's ranking algorithm. The experimental result shows that CT-Rank has the best performance under different temporal textual queries. Furthermore, CT-Rank can support queries and ranking for future time, which is also better than its competitors.

The remainder of the paper is organized as follows. In Section 2, we survey the related work. In Section 3 we introduce the basic idea and design of CT-Rank. Section 4 describes the details about the experiments and the performance evaluation results. Finally, Section 5 concludes the paper and outlines our future work.

2. Related work

2.1. Traditional ranking algorithms

Ranking is one of the core technologies of Web search engines. A lot of ranking algorithms have been proposed so far, which can be classified into three categories.

(1) Link Analysis Based Ranking Algorithms. The first one is the ranking algorithms based on link analysis. The most famous algorithms of this kind are *Pagerank* [1, 2] and HITS [1]. *Pagerank* determines the ranking order of the Web pages according to the number of Web pages that are linked by other pages in the whole Web. The more the linked number of a Web page, the higher its value is. *Pagerank* is an offline algorithm which does not calculate the ranking scores of Web pages during query processing but before this stage. So it is helpful to reduce the response time of query. However, it will lead to a bad sorting result because it ignores the topic relevance between Web pages and user queries. For example, new Web pages will possibly have low ranking scores and will not return to user even if they are mostly topic-related. The HITS algorithm was proposed by Kleinberg at the end of 90's [1]. It assesses the quality of a Web page by two numerical factors, which are content authority (*Authority*) and link authority (*Hub*). The *Authority* of a Web page is related with its referential count in other Web pages (or in other words, its in-link count). A high *Authority* generally means the Web page is frequently referenced in other pages. Similarly, the Hub of a Web page is related with the quality of its hyperlink (out-link). A high Hub means the Web page references many high-quality Web pages. HITS has to compute the *Authority* and *Hub* based on the link relationships between the resulting Web pages and other pages. This makes it difficult to use HITS in a practical application environment.

(2) Online Ranking Models. The second type is based on online ranking model. Unlike the offline *Pagerank* algorithm, the online ranking model computes the ranking scores during the query processing. Such ranking models are Boolean Model [6, 7], Vector Space Model (VSM) [7, 8] and Probability Model [6, 7, 9]. The Boolean Model computes the similarity between Web pages and query by setting the state of every keyword in the query as existing (*true*) or not existing (*false*) in Web pages. The query keywords can be connected by AND, OR and NOT opera et al. in the 1960s [7]. According to this model, every Web page and query is represented as a vector with equal length. The similarities between the query vector and pages' vectors are then measured to determine the relevance between query and Web pages. The Vector Space Model naturally introduces flexibility and ambiguity of retrieval, and makes the retrieval more reasonably. However, there are some problems in this model. For example, the model turns the process of retrieval into the computation of vectors and can not reflect the complex relationships between pages. Besides, there are a large amount of calculations in the algorithm, which will have big impacts on the search performance. The Probability Model sorts Web pages by the probabilities of relevance between the query and Web pages. The probability of relevance is theoretically more strict and reasonable than the Boolean Model. However, it introduces the additional cost of storage and computation. Meanwhile, it has to estimate some parameters in the algorithm, which adds the difficult of the usage of this model. For all the online ranking models, the biggest problem is the huge computation cost and therefore the bad performance of search engines, especially when the amount of Web pages becomes very large.

(3) Relevance Feedback Based Models. The third kind of ranking algorithms are based on the relevance feedback model [10, 11]. The relevance feedback model is one type of self-learning technique. It can automatically adjust user queries based on the previously retrieved results. Thus it is possible for the relevance feedback model to retrieve more needed results and delete those unrelated results. However, the relevance feedback model is not a retrieval model. It needs to be combined with some retrieval model to improve the search efficiency.

2.2. Time-related ranking algorithms

Time is a very important factor in Web search. In recent years, there are also some related researches in time-based Web search and some of them are focused on the ranking issue.

Most of time-related Web search concentrates on Web archive system [12-14]. A Web archive system tries to store and manage historical Web pages and then provide evolutional information of the Web. The history of a Web page is typically captured by versioning technique, i.e., the new version of a Web page is stored with an explicit update timestamp. However, Web archive systems only consider the update timestamps of Web pages. They do not take into account the content time of Web pages, which is much different from the research scope of our algorithm.

In recent years, several researchers have tried to add temporal factor to links and to study ways to find fresh Web pages. The freshness which is the time when a Web page or link was last updated and the activity which is the rate of updates of a page or a link's are considered the most efficient to improve the search engine. Some time-related ranking algorithms are also proposed to put the update time of Web pages into the ranking framework. Those ranking algorithms are not only useful in Web archive systems, but also helpful to improve the ranking effectiveness of traditional search engines. The basic idea so far is to give fresh Web pages higher ranking score and to retrieve the last updated version of Web pages [12, 15-19]. For example, in [19] a time-related ranking algorithm was presented to rank fresher Web pages in front of the old ones. Those ranking algorithms can be summarized as the "*Update-Time-Based Ranking*" algorithms, and we will use this terminology in the following text.

Time-related ranking algorithms were also studied in some specific Web applications. The *TimedPageRank* algorithm [17] was proposed in a Web-based literature searching prototype. It uses the posted time of paper to perform the ranking process. If we map it into a general Web search engine, the posted time of paper can be regarded as the update time of Web page. It can not support queries focusing on the content time. In [20], a temporal search system for business hours was studied, which tried to answer such questions "Which shops are open and in which time are they open". In this system, the time granularity was restricted in hour, e.g. "9:00 AM". Besides, it does not support implicit time, such as Christmas, the National Day. So it is not suitable for general Web search engines.

Also recently, Google has added the *view:timeline* feature to display search results along a timeline, allowing a limited exploration of a hit list. Google also support a range date search as part of the advanced search options. The search results are then filtered based on the date a Web page has been created or last modified. In some cases, this approach can be misleading, because the timestamp is provided by a Web server and may not be accurate. Thus, for search purposes, the time dimension is mainly restricted to the metadata associated with Web pages and does not exploit the temporal information embedded in the pages.

3. The CT-Rank algorithm

3.1. Basic idea

CT-Rank is an extension of traditional ranking algorithm. It considers both text relevance and time relevance of Web page. The basic idea of CT-Rank can be described as follows:

(1) CT-Rank algorithm considers both text relevance and time relevance of Web page when computing the scores of Web pages. The text relevance is defined by the relevance between the keywords in the query and the keywords in Web pages. Time relevance is computed according to the relevance between the time range in the query and the content time in Web pages. The update time of Web pages is used in the extraction of content time.

(2) We map each keyword in a Web page with a specific content time period and then calculate the time-constrained *tf-idf* score of each keyword. Hence, we will construct a set of <*keyword*, *time*, *score*> pairs for each Web page, in which the time represents the most relevant content time of the given keyword.

(3) We use a two-staged design to implement the CT-Rank algorithm. The first offline stage is to construct the <*keyword*, *time*, *score*> pairs for each Web page. The second online stage is to compute the final ranking scores for all the Web pages. In the second stage, we combine three factors, namely the *pagerank* value, the title ranking score, and time-constrained keyword score, to achieve a tradeoff between time relevance and text relevance.

The main difference between CT-Rank and other existing ranking algorithms is that it combines content time and update time of Web page into the ranking algorithm. According to our knowledge, there are few previous works focusing on this direction. Furthermore, our <*keyword*, *time*, *score*> mapping policy also introduces a reasonable solution to integrate text relevance and time relevance into the ranking algorithm.

3.2. The two-stage working process

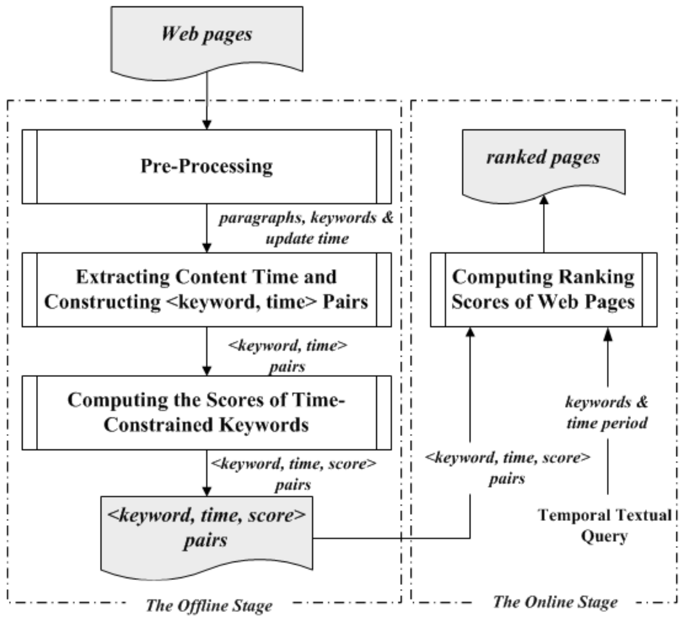

Web pages
Pre-Processing
paragraphs, keywords & update time
Extracting Content Time and Constructing <keyword, time> Pairs
<keyword, time> pairs
Computing the Scores of Time-Constrained Keywords
<keyword, time, score> pairs
<keyword, time, score> pairs
ranked pages
Computing Ranking Scores of Web Pages
keywords & time period
<keyword, time, score> pairs
Temporal Textual Query
The Offline Stage
The Online Stage

Figure 1. The two-stage working process of CT-Rank

Fig.1 shows the two-stage working process of CT-Rank. The offline stage is designed to extract time information (including content time and update time) and keywords of each Web page and construct <*keyword*, *time*, *score*> pairs, where *score* represents the *time*-constrained *tf-idf* score of *keyword*. The online stage is executed in the query processing procedure. It will calculate the final ranking scores for all the result pages.

3.2.1. Preprocessing

The preprocessing step deals with paragraph segmentation and word segmentation. As a result, it outputs the paragraphs, sentences and keywords set of each Web page. It contains the following three steps:

(1) Delete <script>, <span> and other interference labels, and extract the body of the page, then segment paragraphs according to <p>, <br>, <li> and other tags. After this, segment every paragraph into sentences using ".","!","?" and other punctuation marks, then again segment every sentence into smaller sentences according to ",",";"and other punctuation marks. In the preprocessing, we use the Open-source tool ICTCLAS **[**21**]** for the word segmentation. According to statistics, we find that the query morphological features of users usually contains noun, verb, adjective and so on, and rarely contains adverb, interjection, and preposition. So we ignore the words whose morphological features are adverb, interjection, or preposition. Fig.2 shows an example of the extraction result of paragraphs, sentences, and keywords.

(2) Extracting the update time of Web pages based on their meta information provided by the Web server.

(3) Extracting the title of each Web page according to the <title> tag. Then we segment the title into a set of keywords and determine the corresponding time of the title, as called *title time* in the following text. If the title contains no time words, we use the update time of the page as the title time.

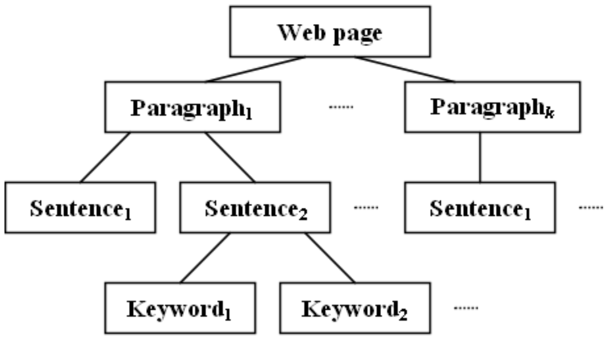


Figure 2. The preprocessed result of a Web page

3.2.2. Extracting content time of Web pages

Time annotation stems from the traditional research on natural language processing (NLP) [22]. Although there is some previous work on the time annotation on text, rare work has been done for that on Web pages. Time annotation on text is usually based on two standards: TIMEX2 [23] and TimeML [24]. The most important difference between Web page and text is that a Web page has some tags. So in this paper, we first eliminate the tags in a Web page, and then apply the traditional time annotation approaches to obtain the content time information in the Web page. For each extracted content time, we record the number of its container sentence, as well as its position in the sentence.

3.2.3 Constructing <*keyword, time*> Pairs

A previous work shown that about 70% query keywords are about time and space [9]. Generally, a time-related query contains several keywords and one or more time words. Moreover, the keywords and the time words in most time-related queries usually imply some relationships which represent users' indeed searching needs. For example, a query "China population statistics 2008" in Google actually means that users want to find China population statistics in 2008. Here, the text keywords "China population statistics" and the time word "2008" in the query have a significant relationship. In general, the relationship between keyword and time words can be represented as a pair <keyword, time>, which indicates that the given keyword is mostly related with the given time.

Current search engines deals with the text keywords and time words individually and ignores the relationship between the keywords and time words. Our CT-Rank algorithm is designed to present a better solution for this problem. We will consider the relationship between keywords and time information when performing the ranking task. For this purpose, we first find the most relevant content time for each keyword, and construct <*keyword*, *time*> pairs. After that, we can computer the time-constrained ranking contribution of each keyword.

The detailed algorithm to construct <*keyword*, *time*> pairs is shown in Fig.3. The input of the algorithm is a paragraph set of the Web page (as shown in Fig.2) and the update time of the Web page.

```
Algorithm Time_Mapping
Input:  $P$ : an ordered paragraph list $< P_1, P_2, ..., P_n >$, where $P_i$ is the ith paragraph in the page.
        $P_i =< S_1, S_2, ..S_m >$, where $S_j$ is the jth sentence in $P_i$. $S_i =< W_1, W_2, ..W_k >$, where
        $W_t$ is the tth keyword in $S_i$.
        $ut$ : the update time of the Web page
Output: $M$ : the mapping list of <keyword, time> pairs
Begin:
  If $S_1 \in P_1$ and $S_1$ contains no time Then
      //initializing the first sentence in the first paragraph
      Set the title time as the time of $S_1$;
  End If
  For Each $P_i$ in $P$, i =1, 2,.. Do
      For Each $S_j$ in $P_i$, j =1, 2,.. Do
          If $S_j$ contains time t Then
              For Each $W_t$ in $S_j$ Do Append <$W_t$, t> into $M$;
          Else
              t ← FindSimilarTime($S_j$, $P_i$);
              For Each $W_t$ in $S_j$ Do Append <$W_t$, t> into $M$;
          End If
      End For
  End For
End Time_Mapping
```

Figure 3. The algorithm to construct <*keyword*, *time*> pairs

The sub-routine *FindSimilarTime*(S_j, P_i) is used when S_j does not contain an explicit time word. In this paper, we first find the most similar sentence of S_j in the whole page, and then use the time contained in that page as the corresponding time of S_j. The similarity between two sentences are defined by their cosine similarity [25] in case that both sentences are represented as vectors. Fig.4 shows the details of the algorithm *FindSimilarTime*.

In Fig.4, the similarity between two sentences, S_1 and S_2, is defined by the cosine similarity function *Sim*(S_1, S_2). Suppose $W = < w_1, w_2, ..w_n >$ is the set of keywords that appear in both S_1 and S_2, $K = < k_1, k_2, ..k_n >$ and $K' = < k_1', k_2', ..k_n' >$ represents the count of each keyword appearing in S_1 and S_2, such that k_1 is, *Sim*(S_i, S_j) is then defined as follows (the measurement of the similarity between two paragraphs is similar, except that all the common keywords in both paragraphs are considered):

$$Sim(S_1, S_2) = \frac{\sum_{i=1}^{n}(k_i \times k_i')}{\sqrt{\sum_{i=1}^{n} k_i^2} \times \sqrt{\sum_{i=1}^{n} k_i'^2}}$$

```
Algorithm FindSimilarTime
Input: $S, P$: $P$ is a paragraph. $S$ is a sentence contained in $P$. $P =< S_1, S_2, ..S_m >$
Output: $t$: the most similar time for $S$ in the page.
Begin:
  If $S \neq S_1$ Then // $S$ is not the first sentence in $P$
      Suppose $S = S_d$, Sim = 0;
      // Find $S_f$, the most similar sentence of $S$ in $P$
      For i = 1 to d – 1 Do
          Sim = max(Sim($S_i$, $S$ ), Sim); f = i;
      End For
      t ← the time contained in $S_f$;
  Else // $S$ is the first sentence in $P$
      // Find $P_f$, the most similar paragraph of $P$ in the page
      Suppose $P = P_d$, Sim = 0;
      For i = 1 to d – 1 Do
          Sim = max(Sim($P_i$, $P$), Sim); f = i;
      End For
      // Find $S_h$, the most similar sentence of $S$ in $P_f$
      For Each $S_j$ in $P_f$, j =1, 2,.. Do
          Sim = max(Sim($S_j$, $S$ ), Sim); h = i;
      End For
      t ← the time contained in $S_h$;
  End If
  Return t;
End FindSimilarTime
```

Figure 4. The *FindSimilarTime* algorithm

3.2.4 Computing the scores of time-constrained keywords

Table 1. Notations in the improved *tf-idf* formula

Notation	Definition
num_{time}	The total number of all the <*keyword*, *time*> pairs in the Web page, in which *keyword* is the same as the given keyword and *time* is the same as the given time.
num_{total}	The total number of all the keywords in the Web page
$pages_{total}$	The total number of Web pages.
$pages_{keyword}$	The number of Web pages which contain the given keyword.

Next, we compute the ranking contribution of each <*keyword*, *time*> pair. In this paper, we use an improved *tf-idf* method to measure the scores of the time-constrained keywords. The term *tf-idf* (term frequency-inverse document frequency) [8] has been widely applied in modern information retrieval. However, traditional *tf-idf* method does not differentiate text keywords and time words, thus it will have low efficiency if we directly use it in time-aware information retrieval.

Given a <*keyword*, *time*> pair in a Web page *P*, unlike the traditional *tf-idf* method which calculates the frequency of key given keyword in the whole page, we now only consider all the keywords which have the same related time as that in the given <*keyword*, *time*> pair. The improved *tf-idf* method is defined by the following formula:

$$score(keyword) \triangleq (num_{time} / num_{total}) \times \log(pages_{total} / pages_{keyword})$$

The notations used in the improved *tf-idf* formula are defined in Table 1.

3.2.5 Compute ranking scores of Web pages

When the user posts a query to the search engine, the final ranking scores of Web pages are computed dynamically. In this paper, the final ranking score of a Web page is calculated based on three factors, namely the improved *tf-idf* value of each keyword, the *Pagerank* score of the Web page, and the title contribution.

(1) The improved *tf-idf* value of each keyword. This factor reflects the time-constrained keyword relevance between user query and Web pages.

(2) The *Pagerank* score. The authority of a link is one of important indicators for Web page ranking, so we take the *Pagerank* scores of every Web page into the ranking framework.

(3) The title contribution. The title of a Web page has an important impact on users' satisfaction, so we set the title contribution of Web pages as a factor of the final ranking score.

Given a set of query keywords, say $< w_1, w_2,..w_n >$, the computation of the final ranking score is based on the following formula:

$$rank(page) \triangleq (\sum_{i=1}^{n} score(w_i)) + pagerank + score(title)$$

where *score*(w_i) is the improved *tf-idf* score of the query keyword w_i in the page, *pagerank* is the *Pagerank* score, *score*(*title*) is the title contribution of the page. The title contribution *score*(*title*) is defined as follows. If the title contains one of query keywords and the query time, we set *score*(*title*) = *score*(*title*) + 1. If the title contains all the keywords of the query and the query time, we set *score*(*title*) = 2 * *score*(*title*) in order to make the page a high ranking priority in the searching results.

4. Performance evaluation

4.1. Experiment setup

To evaluate the performance of CT-Rank, we conduct an experiment on a dataset gathering from Google. For comparison, we choose four algorithms as the competitors of CT-Rank, which are (1) *Pagerank*, (2) Vector Space Model (VSM), (3) Update Time Based Ranking, and (4) Google's ranking algorithm.

The running process is the same as shown in Fig.1. For every Web page in the dataset, we cut the page into words using ICTCALS [21] and filter some stop words and leave other words as keywords of the Web page. Then we extract the content time in the page and construct <*keyword*, *time*, *score*> pairs. After that, we issue four types of temporal textual queries (see Section 4.4) and measure the ranking results of CT-Rank and other four competitor algorithms. We will use two ways to measure the performance of each algorithm (see Section 4.3). The *Pagerank* value for every Web page is got from http://tool.cnzz.cn/pr/Default.asp.

We implement the algorithms using Java under the developing environment ObjectWeb Lomboz. The test machine has an Intel Dual Core Processor, 2GB of main memory, and is running Windows XP Professional with SP3.

4.2. Dataset

In order to prepare the dataset, we first analyze the search log of Sogou (http://www.sougou.com) and choose four kinds of typical temporal textual queries, which are time instant query, time period query, historical query and future query. Then we use those kinds of temporal textual queries to perform search in Google. We choose Google's advanced search option to execute the queries, because Google's advanced search option allows us to specify a time predicate for a query. Finally, we download the results returned by Google as our dataset. The whole dataset contains 6,500 Web pages.

4.3 Subjective evaluation vs. user evaluation

We use two kinds of ways to measure the performance of each algorithm, one is subjective evaluation and the other is user evaluation. In subjective evaluation, we check each returned page and assign an appropriate score according to prepared rules. In user evaluation, we randomly select some users and ask them to rank the searching results of each algorithm.

For subjective evaluation, we prepare four subjective rules to rank the returned results of each algorithm. Each rule is corresponding with a specific ranking score. We finally calculate the average score of each algorithm and compare the performance. The four rules are Very Relevant (score: 3), Relevant (score: 2), Some Relevant (score: 1), and Irrelevant (score: 0).

For user evaluation, each user is allowed to have his or her subjective assessment. For every Web page in the result list, users are asked to check whether a Web page satisfies his (or her) needs, and then they must choose "Yes" or "No" to return the answer. Then we calculate the proportion of "Yes" and "No" for every result list and compare the performance of algorithms.

4.4 Benchmark queries

Table 2. The eleven benchmark queries

No	Type	Keywords	Time Range (yyyy-mm-dd)
1	Instant Query	House Rent Information	2010-1-1
2	Period Query	Discount Information of the Shopping malls in Shanghai	[2009-10-7, 2009-10-14]
3	Period Query	Na Li[*]	[2009-12-1, 2010-1-4]
4	Period Query	Stamp Prices in Beijing	[2006-1-1, 2006-12-31]
5	Period Query	Entrance Test of the People's Bank of China	[2007-1-1, 2008-12-31]
6	Period Query	Train Timetable in Hefei	[2009-10-1, 2009-11-30]
7	Historical Query	Super Girl Champions in China	[2006-1-1, 2006-12-31]
8	Historical Query	Ranking of Universities	[2006-1-1, 2006-12-31]
9	Historical Query	Commissioned Generals of China	[1955-1-1, 1955-12-31]
10	Future Query	Job Positions in Beijing	[2010-1-1, 2010-1-31]
11	Future Query	Movie	[2010-1-1, 2010-1-31]

[*] Na Li is an arising new star in China.

The benchmark queries are based on the analysis on the query log in Sougou (http://www.sougou.com). We use four types of typical temporal textual queries in our experiment:

(1) *Instant query*: The time granularity in our algorithm is set to day, so the time instant query is related with a specific day. For example, "2009-1-1" is a time instant. In our experiment, the time instant queries are restricted in a recent time range.

(2) *Period query*: Time period is a time range through days, months or years, such as 2009-1-1 to 2009-1-3, 2009-1-1 to 2009-2-20, or 2008 to 2009. In our experiment, the time period queries are restricted in a recent time range.

(3) *Historical query*: These types of queries refer to user queries that are focused on a historical time, e.g. "find the commissioned generals in China in 1955". Since Web pages are frequently updated, a historical query may not always return satisfied results. In our experiment, the historical queries are restricted before year 2007.

(4) *Future query*: If a query refers to a future time then it is called future query. Currently, Google does not support future query in its advanced query option. In our experiment, we restrict such queries in a near future time range.

We choose eleven benchmark queries which covers the above four kinds of queries. The query keywords are determined base on the user search logs in Sougou (http://www.sougou.com), so all of the benchmark queries are from real users' needs. Table 2 shows all the eleven benchmark queries used in the experiment, in which there is a time instant query, five period queries, three historical queries, and two future queries.

4.5 Results

4.5.1 Subjective Evaluation Results

For each algorithm, we execute all the queries in Table 2 and collect the average score of each type of query. In addition, we only choose the top ten returned Web pages to measure the performance of each algorithm. This is because the top ten Web pages usually determine the users' satisfaction on a ranking algorithm. The measuring rules have been discussed in Section 4.3, which are *Very Relevant* (score: 3), *Relevant* (score: 2), *Some Relevant* (score: 1), and *Irrelevant* (score: 0). We manually check the returned top ten pages and assign an appropriate score for each page. However, we find that the sequence of each page in the result list have different impacts on the final score. For example, the first page in the results is generally most important to user. For this reason, we assign different weight to each page in the top ten pages, namely the top ten pages sequentially have the weights of 10, 9, 8, 7, 6, 5, 4, 3, 2, 1. We multiple the subjective score of each page with its corresponding weight and get the final score. For example, if subjective scores of the top ten pages returned in a query are $S = (3, 3, 3, 2, 2, 2, 1, 0, 0, 3)$, then we will get the final evaluation score of the query, which is:

$$score = \sum_{i=1}^{10}(11-i)\cdot S_i = 10\cdot 3+9\cdot 3+8\cdot 3+7\cdot 2+6\cdot 2+5\cdot 2+4\cdot 1+3\cdot 0+2\cdot 0+1\cdot 3 = 124.$$

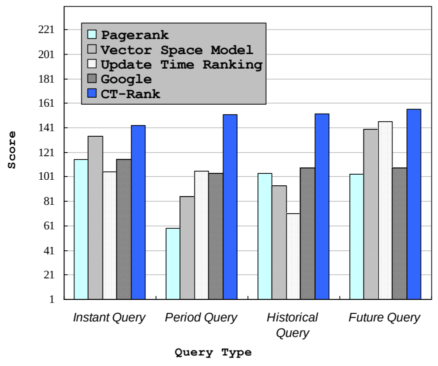


Figure 5. The subjective evaluation results

Fig.5 shows the subjective evaluation results of the CT-Rank algorithm as well as the four competitors. Here the Y axis is the average score of all the same type of queries.

We can see in Fig.5 that the CT-Rank algorithm gets the highest score in all the four kinds of temporal textual queries. Moreover, among all the queries, the CT-Rank algorithm has a relatively stable ranking score.

There are also some results we can find in Fig.5. For the period queries, the score of Vector Space Model and the score of *Pagerank* is relatively low. That is because the time period information is not explicitly expressed in many Web pages, so the Vector Space Model will get a bas result when it uses a directly keyword matching technique to measure the similarity between query and pages. Also for the historical queries, the score of Update Time Ranking is relatively low. That is because no Web pages in the dataset have update time of year 1955. Google exhibits bad performance in future queries, because it only supports update time and treats the content time information of Web pages as keywords.

4.5.2 User evaluation results

In our experiment, we ask ten students in our university to evaluate the result of CT-Rank and Google using the evaluation mentioned in Section 4.3. They are asked to issue the predefined eleven queries in Table 2 and select “Yes” or “No” to evaluate the returned results of each algorithm. The users are blind to what algorithm they are measuring. We get the average proportion of “Yes” in every result to evaluate the performance. The average user satisfaction rate is shown in Fig.6.

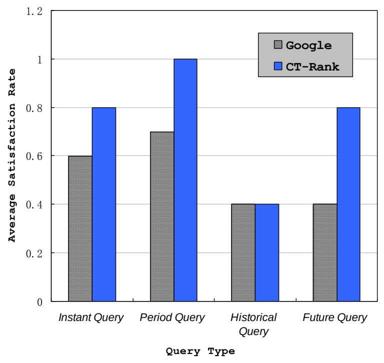
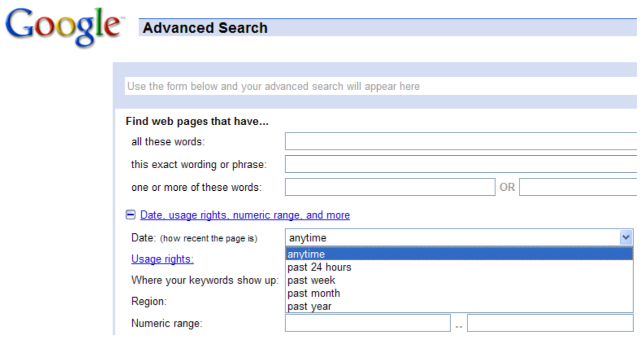


Figure 6. The User evaluation results


Google Advanced Search

Use the form below and your advanced search will appear here

Find web pages that have...

all these words:

this exact wording or phrase:

one or more of these words: OR

⊟ Date, usage rights, numeric range, and more

Date: (how recent the page is) anytime

Usage rights:

Where your keywords show up:

Region:

Numeric range: ..

Figure 7. Google’s historical query in the advanced searchoption

From the figure, users feel that the results of CT-Rank are better than Google’s. Since Google’s advanced search option doest not allow users to input future time condition, as shown in Fig.7, so we are not able to perform future query using Google’s advanced search option. Users can only input the temporal information into Google’s query box as general keywords and run the keyword-based search in Google.

5. Conclusions

In this paper, we introduce the CT-Rank algorithm which is based on the relationship between the content time of Web page and keywords. It presents an appropriate tradeoff between time relevance and keyword relevance. The experimental results show that the CT-Rank algorithm has better performance for temporal textual queries than its competitors including Pagerank, Vector Space Model, Update Time based Ranking, and Google.

6. Acknowledgments

This work is supported by the National High Technology Research and Development Program ("863" Program) of China (Grant No. 2009AA12Z204), the National Natural Science Foundation of China under the grant no. 60776801 and 70803001, the Open Projects Program of National Laboratory of Pattern Recognition (no.20090029), the Key Laboratory of Advanced Information Science and Network Technology of Beijing (no. xdxx1005).

7. References

[1] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In Proc. of WWW, pp. 107-117, 1998.
[2] T. Haveliwala. Topic-sensitive PageRank. In Proc. Of WWW, 2002.
[3] A. Bordin, G. Roberts, et al. Link Analysis Ranking: Algorithms, Theory, and Experiments. ACM Transactions on Internet Technology, Vol. 5(1), pp. 231-297, 2005.
[4] I. Mani, J. Pustejovsky, and R. Gaizauskas (Eds.): The Language of Time. Oxford University Press, 2005.
[5] P. Jin, J. Zhao, L. Yue. Representing Spatiotemporal Information for Web Pages, In Proc. Of NCM, Vol.2, IEEE CS Press, Gyeongju, Korea, 2008
[6] M. Yoshioka, M. Haraguchi. Study on the Combination of Probabilistic and Boolean IR Models for WWW Documents Retrieval. National Institute of Informatics, 2004.
[7] R. Baeza- Yates, B. Ribeiro-Neto. Modern Information Retrieval, ACM Press, 1999.
[8] A. Singhal. Modern Information Retrieval: A Brief Overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol. 24(4), pp. 35-42, 2001
[9] M. Wechsler. The Probability Ranking Principle Revisited. Information Retrieval, 2000.
[10] G Salton, C Buckley, Improving Retrieval Performance by Relevance Feedback. Readings in Information Retrieval, 1997.
[11] T. Qin, T. Liu, X. Zhang. Learning to Rank Relational Objects and its Application to Web Search. In Proc. Of WWW, pp. 407-416, 2008.
[12] E. Deniz, F. Chris, J. P. Terence, Chronica: a Temporal Web Search Engine. In Proc. Of ICWE, pp. 119-120, 2006
[13] C. Dyreson, H. Lin, and Y. Wang, Managing Versions of Web Documents in a Transaction-time Web Server, In Proc. Of WWW, 2004
[14] K. Berberich, S. J. Bedathur, T. Neumann, G. Weikum: A Time Machine for Text Search. In Proc. Of SIGIR, pp. 519-526, 2007
[15] A. Omar, M. Gertz, R. Baeza-Yates, Clustering and Exploring Search Results using Timeline Constructions. In Proc. Of CIKM'09, Hong Kong, China, pp. 97-106, 2009
[16] K. Berberich, M. Vazirgiannis, G. Weikum: T-Rank: Time-Aware Authority Ranking. WAW 2004: 131-142, 2004
[17] P. S. Yu, X. Li, B. Liu: On the Temporal Dimension of Search. In Proc. Of WWW (Alternate Track Papers & Posters) , pp. 448-449, 2004.
[18] N. Sato, M. Uehara, Y. Sakai, Temporal Ranking for Fresh Information Retrieval, In Proc. Of IRAL, pp. 116-123. 2003
[19] A. Jatowt, Y. Kawai, K. Tanaka: Temporal Ranking of Search Engine Results. In Proc. Of WISE, pp. 43-52, 2005

[20] T. Tezuka, K. Tanaka: Temporal and Spatial Attribute Extraction from Web Documents and Time-Specific Regional Web Search System. In Proc. of W2GIS, pp. 14-25, 2004
[21] ICTCLAS,http://www.ictclas.org/.
[22] A. Setzer, R. Gaizauskas, On the Importance of Annotating Event-Event Temporal Relations in Text. In Proc. of LREC, 2002.
[23] TIMEX2 Annotation Standard. in http://fofoca.mitre.org/
[24] TimeML. in http://www.timeml.org/site/index.html
[25] G. Salton , A. Wong , C. S. Yang, A Vector Space Model for Automatic Indexing, Communications of the ACM, vol.18(11), pp. 613-620, 1975