Mining Time-Sensitive Queries and their Sensitivities from Query Log

Jiaqi Li[1], Weiyi Meng[2], Hongyan Liu[3], Shengyun Sun[1], Jun He[1], Zhicheng Dou[1], Xiaoyong Du[1]
[1]Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
School of Information, Renmin University of China, 100872, China
{lijiaqi431,sun_shengyun, hejun, dou, duyong}@ruc.edu.cn
[2]State University of New York at Binghamton
meng@binghamton.edu
[3]School of Economics and Management, Tsinghua University, 100084, China
hyliu@tsinghua.edu.cn

Abstract. In this paper, we present a new concept called *time-sensitive query*. A query is time-sensitive if the set of relevant pages changes when the query is expanded with different temporal terms. Knowing which queries are time-sensitive and what sensitivities they have can be very useful in automatic query expansion for improving search effectiveness and user experience. Existing methods related to time-sensitive query detection are more sudden event oriented and have limitations in detecting other kinds of time-sensitive queries and in determining the time sensitivity. We propose an approach to detecting time-sensitive queries and determining their time sensitivities. Our experiment shows that the approach works well. The accuracy of time-sensitive query detection can reach 92% and the accuracy of time sensitivity calculation can reach 93.12%.

Keywords: time-sensitive query, sensitivity, query log, click-through data

1 Introduction

The increasing size and popularity of the Internet has placed high demands on search engines. When a user submits a query to a search engine, a huge number of results are often returned and many of them are irrelevant to users' information need. It is a big challenge for search engines to determine users' intents correctly and return relevant results to users. In this paper, we define a new type of queries, named *time-sensitive queries*. A query is time-sensitive if the set of relevant results changes when the query is expanded with different temporal terms and its sensitivity is the time interval of change. To meet users' information need, it is necessary to detect time-sensitive queries and determine their sensitivities. As shown in Figure 1, when query "VLDB call for paper" was submitted to the Bing search engine on 10/14/2015, none of the top 6 results was the latest and the most relevant result about conference VLDB 2016. But, if we know "VLDB call for paper" is a time-sensitive query and its sensitivity is "one year", we can build a proper query (e.g., add "2016" to the original query) to find more recent and relevant results.

There are several potential benefits to recognize time-sensitive queries and determine their sensitivities. Specifically, it can positively affect the processing of time-sensitive queries in two different ways. First, an appropriate time (based on the time sensitivity) may be automatically added to a time-sensitive query to help find more recent and relevant results. For the above example, after the temporal term "2016" was added to "VLDB call for paper", the new query led to better results – the top two results were both relevant as shown in Figure 2. Second, it becomes possible for a search engine to form multiple queries with different times, retrieve the relevant results for each query and display the results in reverse chronological order. For example, suppose it is known that Microsoft reports its earning each quarter. Then for query "Microsoft earning report" submitted on 5/31/2016, the search engine may form queries "Microsoft earning report first quarter 2016", "Microsoft earning report fourth quarter 2015", and "Microsoft earning report third quarter 2015", etc. As a result, the search engine can report Microsoft's earnings for the last several quarters in reverse chronological order.

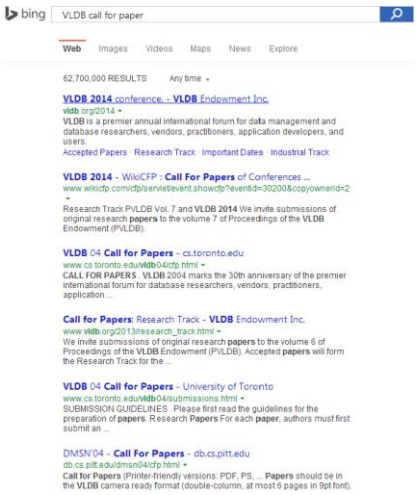

Fig.1. Results of querying "VLDB call for paper"

Another potential benefit is that the method for identifying time-sensitive queries may be used to identify time-sensitive fact statements and time-sensitive questions, which can help identify different truthful statements [1] or correct answers. For example, knowing that "President of United States" is a time-sensitive query may help determine that "Obama was President of United States" is a time-sensitive statement or "Who was President of United States" is a time-sensitive question. Knowing that the time sensitivity of query "President of United States" is 4-year, we can construct different questions by adding years such as "2015", "2011", "2007", … and identify the name of the president of each period.

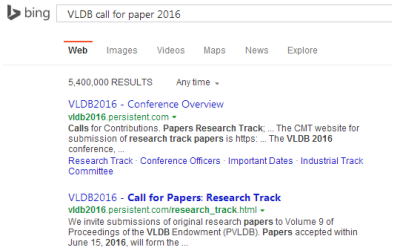

Fig. 2. Results of query "VLDB call for paper 2016"

Our contributions in this work are as follows. (1) We present the concept of time-sensitive query and its sensitivity. (2) We propose a method to detect time-sensitive queries. (3) We develop effective methods to determine the time sensitivity for each time-sensitive query. We conduct experiments on real query log dataset and demonstrate that our proposed methods are highly accurate.

The rest of the paper is organized as follows. In Section 2 we review related work. In Section 3, we introduce the concept of time-sensitive query and its sensitivity and give an overview of our solution. In Section 4 we describe our approach to detecting time-sensitive queries. In Section 5, we provide a detailed description of sensitivity calculation. In Section 6, we report and analyze the experimental results, and finally we conclude the paper in Section 7.

2 Related Work

Dakka et al. studied time-sensitive queries over blogs and news archives [2]. In this work, a query is defined to be time-sensitive if relevant documents for the query are not spread uniformly over time but rather tend to be concentrated in restricted time intervals [2]. The definition of time-sensitive query and the proposed method are more suitable for queries about specific events. In contrast, in our work, a query against the Web is considered to be time-sensitive if there are different sets of relevant documents when the query is associated with different time periods. This definition is more suitable for queries about events that happen periodically.

The works in [3] and [4] incorporate document temporal information into language models to give preference to more recent documents. The authors of [5, 6] studied the distribution of query results over a timeline and used the information to identify significant events. They also attempted to predict the precision of the query results. They proposed a method to automatically classify queries as atemporal, temporally ambiguous, or temporally unambiguous.

In [7], the authors proposed a supervised learning method to automatically classify queries as ambiguous query, broad query and clear query. Similar to works [5] and [6], in [8], time-implicit queries are classified into one of three categories: atemporal, temporal unambiguous and temporal ambiguous by exploiting web snippets, a content-related resource, and exploiting Google and Yahoo! Completion engines that leverage query log resources.

None of the above methods uses query log to identify time-sensitive queries. The works in [9] and [10] discussed using a query log to identify year qualified queries (YQQs), a subset of temporal queries, and use the information to re-rank search results so that more recent results are ranked higher. The method in [9] can be considered as a vastly simplified version of our proposed method. It first identifies queries in the query log that include year information and then the topic parts of these queries are considered as YQQs.

Lee and Kim proposed a method to detecting turning point of a time-sensitive query [11]. They believe each time-sensitive query has a turning point in terms of search volume. But if a time-sensitive query can only be detected after the occurrence of its turning point, it is a bit late. They did not study how to determine the sensitivity of time-sensitive query. Vlachos et al. studied how to determine the periods of a query [12]. The proposed method first creates a time series of the number of times that a query is issued in a day, and then determines the period of the query based on Normalized Discrete Fourier transformation and Power Spectral Density (PSD). To determine the period correctly, the time series about long period queries need to be long enough.

Most existing studies assume that the burst occurrence of a query imply the happening of an event. This is true for event queries, but not true for all time-sensitive queries. In this paper, we give a more general definition of time-sensitive query and propose methods to determine time sensitivity based on query keywords.

3 Problem Definition

Definition 1 (**Time-Sensitive query**) A query submitted to a web search engine is time-sensitive if the set of relevant pages changes when the query is expanded with different temporal terms.

Temporal terms can be in different granularity. For example, 2015, December 2015 and Tuesday are all temporal terms.

Definition 2 (**Sensitivity of a time-sensitive query**) The sensitivity of a time-sensitive query is a measure of the time interval for the set of relevant pages to change.

For example, query "VLDB call for paper" is time-sensitive and its sensitivity is "one year" because a new "VLDB call for paper" page will come out each year. Query "World Cup events" is also time-sensitive with sensitivity "4 years" as World Cup is held every 4 years and each time a new set of events will be posted on the Web. The time sensitivity of query "Microsoft earning report" is "quarterly" as Microsoft reports its earning each quarter of every year.

The information we exploit to identify time-sensitive queries and their sensitivities is click-through data as shown in Table 1. Each record consists of a user's query along with the URL which the user clicked from among the candidates at a specific time. The data set comes from the Bing Search query log excerpt with 68,138,432 queries and 814,243,946 URLs from US users, sampled over one year.

Table 1. A slice of a query log

Time	20130512203658	20130531162339
Query	1990 election	Olympics 2008
Query ID	QID1	QID2
Session ID	SID1	SID2
Click Time	20130512203700	20130531162350
Clicked URL	http://en.wikipedia.org/wiki/Polish_presidential_election%2C_1990	http://hrw.org/campaigns/china/beijing08/

Given a click-through data set, our goal is to detect time-sensitive queries and their sensitivities. Our approach consists of three main steps: preprocessing, detection of time-sensitive queries and calculation of sensitivities. Preprocessing includes stemming [13], stop words removal and synonym recognition [14]. Detection of time-sensitive queries finds potential time-sensitive queries first, and then clusters queries according to query topic.

In order to find time-sensitive queries, we first identify queries that contain a temporal term as well as some non-temporal terms. Each such query consists of two parts: the temporal term and topic part (the non-temporal term part). Later we will show that the topic part of such a query is potentially a time-sensitive query. We will call such queries as potential time-sensitive query containing queries, or simply potential time-sensitive queries. In order to such queries and extract the topic parts and temporal terms from them, we build a time parser for temporal terms. Then we employ a clustering algorithm to place queries into the same group only when their topic parts represent the same concept. Clustering is an important step in our method and we will describe the detail in Section 4.2. Calculation of sensitivities itself includes two main steps: error correction and sensitivity selection. The purpose of error correction is to identify wrong temporal terms and correct them. After error correction, we develop three methods to choose time intervals as the sensitivities of time-sensitive queries. We will introduce the detail in Section 5.

4 Detection of Time-Sensitive Query

4.1 Detecting Potential Time-Sensitive Queries

In order to detect time-sensitive queries, we need to build a time parser. The time parser consists of two parts, a dictionary of temporal terms and a set of time regular expressions. Temporal terms include all seasons (Spring to Winter), all months (January to December), all days in a week (Monday to Sunday), daily, weekly, monthly, bi-monthly, quarterly, semi-annually, annually, decade, century, and abbreviations of temporal terms.

The manually created dictionary of temporal terms is unlikely to cover all the temporal terms. To find time-sensitive queries that contain temporal terms not in the dictionary such as "who won the Indy 500 race in 1977", we also create some temporal regular expressions. These expressions can match years, months, days and their combinations such as "1977", "December 1977", "Dec 21th, 2012" and so on.

Based on the dictionary and time regular expressions, we scan the query log and output all queries that contain one or more temporal terms as the set Q of potential time-sensitive queries. Then, we divide each query in Q into two parts: the topic part, denoted as *topic(Q)*, and the temporal part, denoted as *temp(Q)*. For example, for query "who won the Indy 500 race in 1977", *topic(Q)*= "who won the Indy 500 race in" and *temp(Q)*="1977".

4.2 Topic Clustering

To find time-sensitive queries and determine their sensitivities, we need to group together all queries with the same search intent regardless of the time information. That's why we split each potential time-sensitive query into the topic part and the temporal part. We want to

place query topics into the same cluster if they represent the same concept. The clustering is very challenging. For example, query topic "FIFA World Cup" is equivalent to "World Cup", but "World Cup South Africa" is not equivalent to "World Cup", because the former refers to a specific World Cup, i.e., World Cup 2010, but the latter is not limited to any specific World Cup. In this section we describe how we cluster queries representing the same concept into the same group.

4.2.1 Clustering Algorithm

There are many clustering algorithms to choose from. But to select a proper algorithm that is effective and efficient, there are several challenges. First, the number of clusters of our query log is unknown. It is unreasonable to manually specify the number of clusters in advance. So the algorithm should be able to automatically determine the number of clusters. Second, the click-through data from a search engine are usually huge. Therefore, the clustering algorithm must be efficient and scalable to handle very large data sets within reasonable time and space constraints. Lastly, due to the fact that the query logs increase dynamically, the clustering algorithm should be run incrementally. We choose the density-based clustering method *Incremental DBSCAN* [15] in our work as it meets all of the above requirements. Each cluster obtained by using this method consists of the minimum number of $MinPts$ points, and for every point in the cluster, there exists another point in the same cluster whose distance is less than a distance threshold Eps.

4.2.2 Measuring Similarity of Queries

Similarity measure between queries is another important factor of obtaining satisfactory clustering. In traditional information retrieval (IR), similarity is usually calculated based on keywords. But queries are usually very short and the words are ambiguous. On average a query contains 4.6 words in our data set. Moreover, keywords and meanings do not totally correspond. Users with the same information need may submit different queries to a search engine and select the same URLs from among the candidates offered by the search engine. On the other hand, phrasing the same query, users may click different URLs, that is to say, the same word does not always refer to the same thing. It is hard to determine the user's search intent only based on query content. Suppose a user submits a query "Apple". It's difficult to determine whether the user is interested in something about fruit or information about Apple Company.

In this paper, we measure query similarity based on four types of information: query keywords, user-clicked URLs, query session and knowledge base.

4.2.2.1Similarity Based on Query Keywords

In this measure we use keywords in queries to calculate similarity. Before calculating similarity based on keywords, we need do some preprocessing tasks including stemming [16], stop words removal and synonym recognition. Keywords are words or phrases except for stop words. Usually, the more common keywords two queries share, the more similar they are. Given two queries p and q, we use the Dice's Coefficient [16] to computer the keywords based similarity:

$$S_1 = \frac{|p.keyword \cap q.keyword| \times 2}{|p.keyword| + |q.keyword|} \quad (1)$$

where $p.keyword$ is the set of keywords in the query content of p [17].

4.2.2.2 Similarity Based on User Clicked URLs.

When users submit queries to a search engine, they may click several URLs among the returned results for one query, which are helpful to determine users' intent. Therefore in this measure we calculate similarity of two queries based on URLs clicked by the users for these queries. Again, Dice function [16] is used to calculate this similarity as shown below:

$$S_2(p,q) = \frac{|p.url \cap q.url| \times 2}{|p.url| + |q.url|} \quad (2)$$

In Equation 2, $p.url$ is the set of URLs clicked by a user who submitted query p. Generally, the more common URLs two queries have, the more similar they are.

4.2.2.3 Similarity Based on Sessions.

Usually, users try different queries before they find satisfactory results in a session [18,19]. For example, in a session a user submitted a query "Very Large Data Bases" after searching "VLDB", while the latter is the acronym of the former. Queries in the same session are usually related. Based on this observation, queries in the same session are similar to each other to some extent. For a query q, we use $document_q$ to denote the document that is formed based on all queries in the same session as q. In other words, we concatenate all queries in the same session to form a session document. In this work, sessions are identified based on Session ID recorded in the query log. We define session-based similarity between two queries as the similarity between the session documents of the two queries, as shown in Equation 3.

$$S_3(p,q) = S(document_q, document_q) \quad (3)$$

Here $S(document_q, document_q)$ is the Cosine similarity of the two session documents.

4.2.2.4 Similarity based on Knowledge Base

To measure similarity of two queries based on their semantic, we compute the similarity between their contexts. The context that we use in this paper comes from a large-scale probabilistic semantic network,

known as *Probase* [20] built based on a large web corpus. A term's context is the entire set of concepts that the term belongs to or all the entities that it subsumes in Probase. Given two queries p and q, their similarity, denoted by $S_4(p, q)$, is measured based on their context and cosine similarity [21] .

4.2.2.5 Combination of Multiple Measures

The four types of similarity functions introduced above are based on different types of information. The keyword based function tends to cluster queries with the same or similar keywords. The URLs based function tends to cluster queries that clicked the same URLs. The session based function tends to cluster queries with similar search context. The knowledge base based function tends to cluster queries with similar ontology information. Taking all of the four aspects into consideration, we combine the above similarities linearly as shown in Equation 4.

$$\text{similarity}(\text{p}, \text{q}) = \sum_{i=1}^{4} w_i \times S_i(p, q) \quad (4)$$

Weights w_i (i=1, 2, 3, 4) will be set empirically. Specifically, in our experiments, we test different combinations of the weights to find a good clustering result.

4.3 Filtering Noises

For queries whose topic parts are placed into the same cluster, their temporal parts are also grouped together. We need to determine the sensitivity of each cluster representing the same topic by analyzing the time information. Due to the limitations of the time parser and the clustering method, some queries may be incorrectly recognized as time-sensitive queries. To reduce error, we introduce a filter to throw away clusters that do not have sufficient information for determining time-sensitive queries reasonably reliably. Given a cluster G, if it satisfies all of the following two conditions, we consider it as a true time-sensitive query cluster, and then we can calculate its sensitivity.

1) The group of the temporal parts of a cluster must contain a sufficient number of different temporal terms. In our experiments, we set this number as 3.
2) To ensure the reliability of data, the queries in the same cluster must come from at least two different sessions.

Table 2 gives an instance of a time-sensitive query cluster.

Table 2. A time-sensitive query cluster

Query	Session ID	Topic	Time
thanksgiving 2012	2429BF04BDB3666C1 8F3BDF9BCB666011	thanksgiving	2012
thanksgiving 2011	2429BF04BDB3666C1 8F3BDF9BCB666011	thanksgiving	2011
when is thanksgiving 2013	1E86A334D145659D1 E32A749D05665411	when is thanksgiving	2013
thanksgiving 2013	0595531A8A616C662 4A657538B706C111	thanksgiving	2013

Now all topic parts in the remaining clusters are recognized as time-sensitive queries.

5 Calculation of Time Sensitivity

As mentioned in Section 3, a query is time-sensitive if the set of relevant pages changes when the query is expanded with different temporal terms. After the detection of time-sensitive queries, we have obtained clusters of queries such that queries in the same cluster are about the same topic and each cluster contains multiple temporal terms. Next, we need to determine the time sensitivity of each cluster based on the temporal terms that appear in the queries of each cluster. This task is challenging because there are many kinds of temporal terms in the search log and these temporal terms may be in different granularities. Based on the definition of the time sensitivity of a time-sensitive query, which is a measure of the time interval for the set of relevant pages to change, we classify time-sensitive queries into two categories: *periodic* time-sensitive queries and *irregular* time-sensitive queries.

A time-sensitive query is periodic if the set of relevant web pages changes in regular time interval. A query is an irregular time-sensitive query if it is time-sensitive but not periodic. For example, query "Oscars" is a periodic time-sensitive query, because a new "Oscars" page will come out each year. However, the query "economic crisis" is time-sensitive but not periodic, as economic crisis does not happen at a regular interval. Given the two potential benefits to recognize time-sensitive queries and determine their sensitivities (see Section 1), in this paper we focus on how to determine the sensitivity of periodic time-sensitive queries, after detecting both periodic and irregular time-sensitive queries.

Before sensitivity calculation, we should clean the data to unify the format of time expressions and remove some noises. People use different formats to express time and queries may include wrong time information. Unifying the temporal term format involves following

several steps:

1) In this work, we are interested in temporal terms at the granularity of day or larger. Therefore, terms such as "second", "minute", "hour", "3:30", "am", "pm", "morning", "noon", "afternoon", "evening", "night" are removed.
2) Abbreviations of temporal terms are restored to their full names. For example, "Jan" is replaced by "January" and "Fri" by "Friday".
3) Temporal terms in different formats are mapped to the same format. We use mm/dd/yyyy as the uniform format for date. So "2011/06/03" is converted to "06/03/2011" and "January 5 2011" is converted to "01/05/2011". If a temporal term contains the day of a week, we keep it at the end of the uniform format such as "01/03/2011 Monday".

5.1 Error Correction

When phrasing queries, users may sometimes input wrong temporal information. For example, a user may submit query "2005 Summer Olympics" to a search engine, but there wasn't any Summer Olympics in the year 2005. Such wrong temporal information adds noise to our data set and increases the difficulty to obtain correct time sensitivity. We use the information of clicked URLs to filter out these noises. Our rule is very simple: if the URL and the topic part of the query have at least one common keyword, we will use the temporal term in the URL to replace the temporal term in the query. As shown in Table 3, a query "2005 summer Olympics" was submitted to the search engine, but a URL related to "2008 summer Olympics" was clicked. The query "2005 summer Olympics" and the clicked URL have two common words, so we use the year 2008 in the URL to replace 2005. Similarly, query "1990 Presidential Election" is converted into "1984 Presidential Election".

Table 3. Examples of error correction

Query	1990 Presidential Election	2005 summer Olympics
URLs	http://en.wikipedia.org/wiki/U.S._presidential_election%2C_1984	http://www.2008-summer-olympics-beijing-china.com/blog
After error correction	1984 Presidential Election	2008 summer Olympics

5.2 Calculating Sensitivity

For each cluster of time-sensitive queries, we assume the time sensitivities of these queries are the same. Although through the error correction step some noise has been removed, some may still remain. Moreover, due to the limitation of the search log used, it's impossible to have queries with every time interval of a periodic query. For example, for Olympic Games, we may only have queries with temporal terms 1920, 1932, 1960, 1962, 1964, etc. in a query log. From these time information, we can get several different time intervals. How to find the correct time interval is a challenging issue.

To facilitate the selection of correct time intervals, we calculate the frequency of each temporal term, which is defined below.

Definition 3 (**Frequency of temporal term**) Given a cluster G of queries and a temporal term t of a particular time granularity related to queries in G, the frequency of t is defined as the number of queries in G which contain the temporal term t.

Intuitively the larger the frequency is, the more likely the temporal term is correct. After calculating the frequency f_i of each temporal term t_i, in a cluster, we obtain a set of term-frequency pairs. We then sort these pairs in ascending order of the temporal terms in chronological order. For example, term "01/03/2011 Monday" is sorted after "01/01/2011 Saturday". The sequence is called *term sequence with frequency*, denoted by FP.

$$FP =< (t_1, f_1), (t_2, f_2), \dots, (t_n, f_n) > \quad (5)$$

Removing frequency of each temporal term, the remaining sequence is called *term sequence*, denoted by $T =< t_1, t_2, \dots, t_n >$.

Temporal term frequency can be calculated at different granularity levels. For the "year" level (this would include "yearly", "4-yearly", etc.), we keep only the year information in each temporal term and adjust the frequency information accordingly. For example, suppose we have two pairs (06/01/2011, 10) and (2011, 30). The former will be converted to (2011, 10) at first and then merged with the (2011, 30), yielding a new pair (2011, 40). Similarly, for the "month" level, we will ignore "year" only information and convert any finer granularity temporal terms into the month level terms. For example, 05/07/2011 (May 7, 2011) will be converted to 05/2011 (May 2011) and the frequencies of the temporal terms (including converted terms) corresponding to the same month will be added together as the frequency for the month. For each granularity level, we generate a sequence of "gap pairs". Let $FP = < (t_1, f_1), (t_2, f_2), \dots, (t_n, f_n) >$ be the term sequence of n term-frequency pairs of a given granularity level. We generate an interval sequence of n–1 time interval frequency pairs as follows.

$$GP =< (g_1, gf_1), (g_2, gf_2), \dots, (g_{n-1}, gf_{n-1}) > \quad (6)$$

Here, $g_i = \text{gap}(t_{i+1}, t_i)$ and $gf_i = \min\{f_{i+1}, f_i\}$, $i \in [1, \text{n} - 1]$, where $\text{gap}(t_{i+1}, t_i)$ is the gap between two consecutive temporal terms and gf_i is the number of times this gap has occurred (i.e., gap frequency). For example, if the time granularity is month, the gap should be the number of months between two consecutive temporal terms. For the same time interval, we merge them by adding their frequencies together.

We assume one of the time intervals in *GP* is correct and is the sensitivity we look for. We try three different strategies to determine the correct sensitivity: *Minimum interval method, Most Frequent interval method* and *Edit-distance method*. Simply, the Minimum Interval method considers the minimum time interval as the correct sensitivity as defined in Formula (7), while the Most Frequent Interval method considers the most frequent time interval to be the correct sensitivity as defined in Formula (8), where $|gf_i|$ is the number of occurrences of g_i in *GP*.

$$Sensitivity_{min} = \min(g_i), i \in [1, n-1] \quad (7)$$

$$Sensitivity_{frequent} = \text{argmax}_{g_i}|gf_i| \quad (8)$$

In the Edit-distance method, for each time interval in *GP*, we calculate a measure called *edit distance*.

Definition 4 (**Edit distance**) Given a term sequence $T =< t_1, t_2, \ldots, t_n >$. and a time interval $g=t_{i+1}-t_i$, we construct a potential temporal term sequence S consisting of terms starting from t_{i+1} and t_i, creating new terms with increments g and $-g$ respectively, ending with term not greater than t_n and not less than t_1, as shown below.

$$S(\text{g}) =< s_1, \ldots, (t_\text{i} - g), t_\text{i}, t_{\text{i}+1}, (t_{\text{i}+1} + g), \ldots, s_m > \quad (9)$$

where $s_\text{m}=t_{i+1}+k\times g$ and $s_m \leq t_n$, $s_1=t_i-l\times g$ and $s_1 \geq t_1$, k and l are natural numbers. Comparing S with T, let N be the number of terms added to T and removed from T in order to get S. The edit distance of time interval g, $ED(g)$, is calculated by Equation 10.

$$ED(g) = \frac{N}{|\text{S}|} \quad (10)$$

Ideally, if we have enough data, edit distance for the right time interval should be zero. Therefore, the smaller the edit distance is, the more likely the time interval is correct. Therefore, the sensitivity is computed based on the formula below.

$$Sensitivity_{editdistance} = \text{argmin}_{g_i} ED(g_i) \quad (11)$$

The intuition behind this method is that if time interval g is correct (i.e. the sensitivity of the query), the potential temporal term sequence $S(g)$ is the correct term sequence. Then the term sequence T obtained from search log should be more similar to $S(g)$. In an ideal situation, they should be identical. Edit distance can be regarded as a similarity measure. The smaller the distance, the closer T is to $S(g)$, the more likely g is correct. On the other hand, if $S(g)$ is a fake term sequence, relatively more edit operations should be done to change term sequence T into $S(g)$, so the distance becomes larger, indicating S is more likely to be wrong.

For example, for term sequence with frequency *FP*=<(1920, 2), (1932, 1), (1960, 2), (1962, 2), (1964, 2), (1968, 6), (1976, 10) >, we can get interval sequence *GP*= {(2, 4), (4, 2), (8, 6), (12, 1), (28, 1)}. For time interval 4, we construct the potential temporal term sequence *S*(4): *S*(4)={1920, 1924, 1928, 1932, 1936, 1940, 1944, 1948, 1952, 1956, 1960, 1964, 1968, 1972, 1976}. Comparing *S* with the term sequence of *FP*, *T*=<1920, 1932, 1960, 1962, 1964, 1968, 1976>, 9 temporal terms are added and 1 temporal term is removed, so the edit distance is 10/15= 0.667. Similarly, for the same term sequence, the edit distance of 2-year interval, 8-year interval, 12-year interval and 28-year interval are 0.759, 0.875, 1.2 and 2.5, respectively, as shown in Table 4. Among all the edit distances, 0.667 is the smallest value. Therefore, we consider 4-year to be the correct sensitivity.

Table 4. Edit distance

Time interval	Number of edits	|S|	Edit distance
2	22	29	0.759
4	10	15	0.667
8	7	8	0.875
12	6	5	1.2
28	5	2	2.5

6 Experiments

6.1 Detection of Time-Sensitive Queries

The data set on which we conduct experiment comes from the Bing Search query log excerpt with 68,138,432 queries and 814,243,946 URLs from US users, sampled over the year 2013.

Based on the dictionary and time regular expressions we build, we scan the query log and output all queries that contain one or more temporal terms as the set of potential time-sensitive queries, which contains 1,214,237 queries and is the input to the time-sensitive query detection method.

After clustering the queries using a combination of multiple similarity measures, we obtain 26,677 clusters and after filtering out the noise we obtain 2,985 clusters finally. Because each similarity measure focuses on a specific aspect, the results of clustering based on different similarity measures are very different from

each other. At the same time, the results of clustering affect sensitivity calculation directly. If we place a query into a cluster wrongly, it will add noise to this cluster and increase the chance of getting a wrong sensitivity. Similarly, if we place queries that should belong to the same cluster into different clusters, it will increase data sparseness, which makes determining sensitivity more difficult. In our experiments, we compare the accuracy of time-sensitive query detection based on different similarity measures.

We randomly selected 100 clusters from each clustering results and the accuracy is defined as the percentage of clusters that represent time-sensitive queries. The accuracy is shown in Table 5.

Table 5. Time-sensitive query detection

Similarity Based on	#clusters before filtering noise	#clusters after filtering noise	Accuracy
keywords	38378	1478	76%
clicked URLs	53294	1426	84%
sessions	12280	1557	77%
Knowledge base	38603	3749	86%
Combination	26677	2985	92%

It's easy to see that similarity based on keywords has the lowest accuracy. The keyword based similarity measure tends to cluster queries with the same words into the same group. It is hard to determine a user's search intent only based on analysis of query content, because the same word may have multiple meanings and different words may have the same/similar meaning. Users with the same information need may submit different queries to a search engine and two queries with the same content do not always express the same search intent. The knowledge base based similarity, which tends to cluster queries with similar ontology into the same group, has the highest accuracy among the four similarity measures. As different similarity measures solve the problem from different angles, taking all the four strategies into consideration and combining the four similarities, we get the best performance. Therefore, sensitivity calculation is based on this result in the following discussions.

6.2 Calculation of Time Sensitivity

To show the effectiveness of our proposed methods, we compare their performance in terms of accuracy with the Power Spectral Density (PSD) method proposed in [12]. For a query q, PSD first creates a time series, with each element being the number of times that the query is issued in a day, and then determines the period (i.e. time sensitivity) of the query based on Normalized Discrete Fourier transformation and Power Spectral Density (PSD). To the best of our knowledge, this is the only work that proposed method for time sensitivity calculation. Although many research works studied how to detect and use time-sensitive queries, they don't focus on determining time sensitivity.

We obtained 2,985 query clusters for our data set. For each cluster of queries, we assume that the time sensitivity of these queries is the same. Before sensitivity calculation, we clean the data to unify the time format, remove some noise and correct the error of queries as mentioned in Section 5. Temporal term frequency can be calculated at different granularity levels. In our experiments, we first focus on the granularity of "year". Given a cluster G, if it meets the following two conditions, we consider it a cluster of queries with the granularity of "year".

1) Cluster G contains queries with at least two different 4-digit years.
2) The proportion of the queries that contain 4-digit year information in the cluster is higher than 60%.

After this step, we obtain 477 clusters with year granularity.

As the query log used our experiments is one year long, the year level time-sensitive queries are sparse and cannot offer enough temporal terms to reliably determine whether a query is a periodic or irregular time-sensitive query. After manual verification, 340 clusters contain periodic queries. Then we calculate sensitivity for each of the 340 clusters using the proposed three methods separately. If the data set covered several years, we could determine whether a cluster contains periodic queries using some existing periodicity mining methods [22-24]. The accuracy comparison of the three methods and PSD is shown in Table 6. For the PSD method, errors within 30 days are ignored. We can see that the Edit-distance performs best among all the methods. PSD has a very poor performance, which may be due to its dependence on the quality of the dataset, especially the time span of the query log. One-year search log is not enough for this method to detect year level time-sensitive queries.

Table 6. Accuracy of time sensitivity calculation

	Minimum interval	Most Frequent interval	Edit-distance	PSD
Accuracy	91.98%	90.83%	93.12%	3.0%

There are different kinds of time-sensitive queries

among the 340 clusters of queries, including some festivals like "Thanksgiving" and "Easter", some awards like "Oscars" and "Grammys", some sports events like "World cup" and "Olympics", and some events like "American Election" and so on. Table 7 lists several cases of time-sensitive queries and their sensitivity values calculated by different methods. As seen from this table, the first query gets its right sensitivity by all the three methods, the second query gets its right sensitivity by two of them: Most Frequent Interval and Edit-distance, the third query only gets its right sensitivity by Edit-distance method, while we cannot get the right answer for the last one by using any of the three methods, as our data is sparse and cannot offer enough temporal terms to determine the correct time sensitivity.

Table 7. Example queries and their sensitivities

	Minimum Interval	Most Frequent Interval	Edit-distance
CIKM	1	1	1
Summer Olympics	1	4	4
US presidential election	1	2	4
World cup	1	8	8

Our proposed methods can also be used to determine the time sensitivity of other granularity queries. To show this, we conduct experiments on month granularity queries. To identify month granularity time-sensitive queries, similar to year level processing, we use the following two criteria:

1) A cluster contains queries with at least two different months;

2) The proportion of the queries that contain month information in the cluster is higher than 60%.

After this filtering and manual verification, we find nine month-periodic queries. The comparison of different methods is shown in Table 8. For PSD seven days are permitted errors. This time all of our proposed methods have the correct result, and PSD's accuracy increases to 33.3%, much higher than that of year-level sensitivity. PSD's accuracy is still very low. Besides the reason of not having enough long query log, another reason may be the assumption behind the method that the time of burst query is the happening time of corresponding event, which is only right for news event queries, but not for all time sensitive queries.

Table 8. Accuracy of time sensitivity calculation for month granularity queries

	Minimum interval	Most Frequent interval	Edit-distance	PSD
Accuracy	100%	100%	100%	33.3 %

7 Conclusions

In this paper, we proposed a new problem, i.e., time-sensitive query detection and sensitivity determination. A query submitted to a web search engine is time-sensitive if the set of relevant pages changes when the query is expanded with different temporal terms. The sensitivity of a time-sensitive query is a measure of the time interval for the set of relevant pages to change. Detecting time-sensitive query and its sensitivity is helpful to improve search engines' result relevance and enhance user search experience. We proposed a three-step method (preprocessing, detection of time-sensitive queries and calculation of sensitivities) to find out time-sensitive queries and determine their sensitivity values based on click-through data. In the step of time-sensitive query detection, we proposed different similarity measures for effective clustering and the accuracy of time-sensitive query detection can reach 92% for a real click-through data set. We also proposed three techniques (Minimum interval method, Most Frequent interval method and Edit-distance method) to obtain time sensitivity. Among the three methods, Edit-distance performs the best and its accuracy for sensitivity calculation reached 93.12%.

References

1. Li, X., W. Meng, and C. Yu, *T-verifier: Verifying truthfulness of fact statements*, in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering* 2011, IEEE Computer Society. p. 63-74.
2. Dakka, Wisam, Luis Gravano, and Panagiotis G. Ipeirotis. "Answering general time-sensitive queries." *Knowledge and Data Engineering, IEEE Transactions on* 24.2 (2012): 220-235.
3. Zhang, D., et al., *Time-Sensitive Language Modelling for Online Term Recurrence Prediction*, in *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*2009, Springer-Verlag: Cambridge, UK. p. 128-138.
4. Li, X. and W.B. Croft, *Time-based language models*, in *Proceedings of the twelfth international conference on Information and knowledge management*2003, ACM: New Orleans, LA, USA. p. 469-475.

5. Jones, R. and F. Diaz, *Temporal profiles of queries.* ACM Trans. Inf. Syst., 2007. 25(3): p. 14.
6. Diaz, F. and R. Jones, *Using Temporal Profiles of Queries for Precision Prediction.* Sigir, 2004: p. 18-24.
7. Song, R., et al., *Identifying ambiguous queries in web search*, in *Proceedings of the 16th international conference on World Wide Web*2007, ACM: Banff, Alberta, Canada. p. 1169-1170.
8. Ricardo Campos, Mário Jorge Alípio, Gaël Dias. *Using Web Snippets and Web Query-logs to Measure Implicit Temporal Intents in Queries. 2nd Workshop on Query Representation and Understanding of the 34th ACM Annual SIGIR Conference (SIGIR 2011),* Jul 2011, Pekin, China. pp.4 Pages, 2011.
9. Zhang, R., et al., *Search result re-ranking by feedback control adjustment for time-sensitive query*, in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*2009, Association for Computational Linguistics: Boulder, Colorado. p. 165-168.
10. Metzler, D., et al., *Improving search relevance for implicitly temporal queries*, in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*2009, ACM: Boston, MA, USA. p. 700-701.
11. Lee S E, Kim D. *A click model for time-sensitive queries[C]//Proceedings of the 22nd international conference on World Wide Web companio*n. 2013:147-148.
12. Michail Vlachos, Chris Meek, Zografoula Vagena, Dimitrios Gunopulos. *Identifying Similarities, Periodicities and Bursts for Online Search Queries*, SIGMOD2004.
13. Porter, M.F., *An algorithm for suffix stripping*, in *Readings in information retrieval*, J. Karen Sparck and W. Peter, Editors. 1997, Morgan Kaufmann Publishers Inc. p. 313-316.
14. Miller, G.A., *WordNet: a lexical database for English.* Commun. ACM, 1995. 38(11): p. 39-41.
15. Ester, M., et al., *Incremental Clustering for Mining in a Data Warehousing Environment*, in *Proceedings of the 24rd International Conference on Very Large Data Bases*1998, Morgan Kaufmann Publishers Inc. p. 323-333.
16. Dice, L. R. (1945). *Measures of the amount of ecologic association between species*. Ecology, 26(3), 297-302.
17. Wen, J.-R., J.-Y. Nie, and H.-J. Zhang, *Clustering user queries of a search engine*, in *Proceedings of the 10th international conference on World Wide Web*2001, ACM: Hong Kong, Hong Kong. p. 162-168.
18. Cao, H., et al., *Context-aware query suggestion by mining click-through and session data*, in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*2008, ACM: Las Vegas, Nevada, USA. p. 875-883.
19. Liao, Z., et al., *Mining Concept Sequences from Large-Scale Search Logs for Context-Aware Query Suggestion.* ACM Trans. Intell. Syst. Technol., 2011. 3(1): p. 1-40.
20. Wu, Wentao, et al. *Probase: A probabilistic taxonomy for text understanding. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data.* ACM, 2012.
21. Li, Peipei, et al. *Computing term similarity by large probabilistic isa knowledge. Proceedings of the 22nd ACM international conference on Conference on information & knowledge management.* ACM, 2013.
22. Ma, S. and J.L. Hellerstein. *Mining Partially Periodic Event Patterns With Unknown Periods*. In *Proc. ICDE*. 2000.
23. Yang, J., W. Wang, and P.S. Yu, *Mining Asynchronous Periodic Patterns in Time Series Data.* Knowledge & Data Engineering IEEE Transactions on, 2003.15(3): p. 613-628.
24. Yang, Y., et al., *Discovery of Periodic Patterns in Sequence Data: A Variance-Based Approach.* Informs Journal on Computing, 2012. 24(3): p. 372-386.