Determining Time of Queries for Re-ranking Search Results*

Nattiya Kanhabua and Kjetil Nørvåg

Dept. of Computer Science,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract. Recent work on analyzing query logs shows that a significant fraction of queries are temporal, i.e., relevancy is dependent on time, and temporal queries play an important role in many domains, e.g., digital libraries and document archives. Temporal queries can be divided into two types: 1) those with temporal criteria explicitly provided by users, and 2) those with no temporal criteria provided. In this paper, we deal with the latter type of queries, i.e., queries that comprise *only keywords*, and their relevant documents are associated to particular time periods not given by the queries. We propose a number of methods to determine the time of queries using temporal language models. After that, we show how to increase the retrieval effectiveness by using the determined time of queries to re-rank the search results. Through extensive experiments we show that our proposed approaches improve retrieval effectiveness.

1 Introduction

An enormous amount of information is stored in the form of digital documents, examples include web pages harvested and stored in web archives as well as newspaper articles stored in news archives. Information in such document repositories are useful for both expert users, e.g., historians, librarians, and journalists, as well as for students and other people searching for information needs. However, when searching in such *temporal document collections*, it is difficult to achieve high accuracy using simple keyword search because the contents are strongly time-dependent; documents are about events that happened at a particular time period. In addition, accesses to the contents are time-dependent too, i.e., time is a part of the information needs represented by temporal queries.

In previous work [2, 10], searching temporal document collections has been performed by issuing temporal queries composed of keywords, and the creation or update date of documents (called temporal criteria). In that way, a system narrows down the results by retrieving documents according to both text and temporal criteria. Temporal queries can be divided into two categories: 1) those with temporal criteria explicitly provided by users [2, 10], and 2) those with no temporal criteria provided. An example of a query with temporal criteria explicitly provided is "the U.S. Presidential election 2008", while that of a query without temporal criteria provided is "Germany FIFA

* This work has been supported by the LongRec project, partially funded by the Norwegian Research Council.

World Cup". However, for the latter example, a user's temporal intent is *implicitly* provided, i.e., referring to the world cup event in 2006. As mentioned in [1], an analysis of web user query log shows that 1.5% of queries are explicitly provided with temporal criteria [11], i.e., containing temporal expressions, while about 7% of web queries have temporal intent implicitly provided [9].

In this paper, we focus on implicit temporal queries, i.e., temporal queries that comprise *only keywords*, and where relevant documents are associated to particular time periods that are not given by the queries. Through a novel approach for determining the time of queries, or implicit temporal intent, using temporal language models, we are able to increase the retrieval effectiveness by using the determined time of queries to re-rank search results. Thus, the main contributions of this paper are: 1) the first study on how to determine the time of queries without temporal criteria provided, as well as techniques for determining this time, 2) a study on how to incorporate the determined time of queries into the re-ranking search results, and 3) an extensive evaluation of our approaches for determining the time of queries, as well as of re-ranking search results using the time of queries. It should be noted that our approach is language-independent: the only requirement is the availability of a temporal document collection in the query language, such a corpus can be easily obtained from, for example, a news archive.

The organization of the rest of the paper is as follows. In Sect. 2, we give an overview of related work. In Sect. 3, we outline our of document and query models. Then, we explain the use of temporal language models for document dating. In Sect. 4, we present our approaches to determining the time of queries without temporal criteria provided. In Sect. 5, we describe how to use the determined time to improve the retrieval effectiveness. In Sect. 6, we evaluate our proposed query dating, and re-ranking methods. Finally, in Sect. 7, we conclude and outline our future work.

2 Related Work

Recently, a number of papers have described issues of temporal search [2, 10, 13]. In the approaches described in [2, 10], a user explicitly specifies time as a part of query. Typically, such a temporal query is composed of query keywords and temporal criteria, which can be a point in time or a time interval. In general, temporal ranking can be divided into two types: approaches based on *link-based analysis* and *content-based analysis*. The first approach studies link structures of a document and uses this information in a ranking process, whereas the second approach examines the contents of a document instead of links. In our context, we will focus on analyzing contents only because information about links is not available in all domains, and content-based analysis seems to be more practical for a general search application. Temporal ranking exploiting document contents and temporal information are presented in [4, 5, 8, 12, 13].

In [8], Li and Croft incorporated time into language models, called time-based language models, by assigning a document prior using an exponential decay function of a document creation date. They focused on recency queries, such that the more recent documents obtain the higher probabilities of relevance. In [4], Diaz and Jones also used document creation dates to measure the distribution of retrieved documents and create the temporal profile of a query. They showed that the temporal profile together with the

contents of retrieved documents can improve average precision for the query by using a set of different features for discriminating between temporal profiles. In [13], Sato et al. defined a temporal query and proposed ranking taking into account time for fresh information retrieval. In [5] an approach to rank documents by freshness and relevance is presented. In [12], Perkiö et al. introduced a process of automatically detecting a topical trend (the strength of a topic over time) within a document corpus by analyzing the temporal behavior of documents using a statistic topic model.

Dating of documents has been previously studied by de Jong et al. [3], and their approach later extended by Kanhabua and Nørvåg [6]. However, dating short queries and employing the time in ranking has to our knowledge not been performed before.

The most related work to this paper is [1, 9]. Berberich et al. [1] integrated temporal expressions into query-likelihood language modeling, which considers uncertainty inherent to temporal expressions in a query and documents, i.e., temporal expressions can refer to the same time interval even they are not exactly equal. The work by Berberich et al. and our work is similar in the sense that both incorporate time into a ranking in order to improve the retrieval effectiveness for temporal search, however, in their work, the temporal criteria are explicitly provided for a query. Metzler et al. [9] also consider implicit temporal needs in queries. They proposed mining query logs and analyze query frequencies over time in order to identify strongly time-related queries. In addition, they presented a ranking concerning implicit temporal needs, and the experimental results showed that their approach improved the retrieval effectiveness of temporal queries for web search. Rather than relying on user query logs, we propose an alternative for determining the time of queries from the contents.

3 Preliminaries

In this section, we first briefly outline our document and query models. Then, we explain the basic approach to document dating using temporal language models.

3.1 Temporal Document Model

In this paper, a document collection contains a number of corpus documents defined as $C = \{d_1, \ldots, d_n\}$. A document d_i can be seen as bag-of-word (an unordered list of terms), and a creation or updated date. Note that, d_i can also be associated to temporal expressions containing in the contents. However, temporal expressions will not be studied in this paper. Let $\mathit{Time}(d_i)$ be a function that gives a creation or updated date of d_i, so d_i can be represented as $d_i = \{\{w_1, \ldots, w_n\}, \mathit{Time}(d_i)\}$. If C is partitioned wrt. a time granularity of interest, the associated time partition of d_i is a time interval $[t_k, t_{k+1}]$ containing $\mathit{Time}(d_i)$, that is $\mathit{Time}(d_i) \in [t_k, t_{k+1}]$. For example, if we partition C using the *1-month* granularity and $\mathit{Time}(d_i)$ is 2010/03/05, the associated time partition of d_i will be $[2010/03/01, 2010/03/31]$.

3.2 Temporal Query Model

We define a temporal query q as composed of two parts: keywords q_{word} and temporal criteria q_{time}, where $q_{\mathit{word}} = \{w_1, \ldots, w_m\}$, and $q_{\mathit{time}} = \{t'_1, \ldots, t'_l\}$ where t'_j is a time

interval, or $t'_j = [t_j, t_{j+1}]$. In other words, q contains uncertain temporal intent that can be represented by one or more time intervals. We can refer to q_{word} as topical features, and q_{time} as temporal features of q. Hence, our aim is to retrieve documents about the topic of query where their creation dates are corresponding to time criteria.

Recall that temporal queries can be divided into two types: 1) those with temporal criteria explicitly provided by a user, and 2) those with no temporal criteria provided. An example of the first type is "Summer Olympics 2008" where the user interests in documents about "Summer Olympics" written in 2008. In this case, q_{time} is equal to $\{[2008/01/01, 2008/12/31]\}$ given the *1-year* time granularity. Queries in the second type can be implicitly associated with particular time especially queries related to periodic, or outbreak events. The query "Boxing Day tsunami" is associated with the year "2004", $q_{time} = \{[2004/01/01, 2004/12/31]\}$, and the query "the U.S. presidential election" can be associated with the years "2000", "2004", and "2008", so that $q_{time} = \{[2000/01/01, 2000/12/31], \ldots, [2008/01/01, 2008/12/31]\}$. When the time q_{time} is not given explicitly by the user, it has to be determined by the system, as will be described later in this paper.

3.3 Temporal Language Models

The document dating approach is based on the *temporal language model* presented in [3], which is a variant of the time-based model in [8]. The idea is to assign a probability to a time partition according to word usage or word statistics over time.

A normalized log-likelihood ratio [7] is used to compute the similarity between two language models. Given a partitioned corpus, it is possible to determine the timestamp of a non-timestamped document d_i by comparing the language model of d_i with each corpus time partition p_j using the following equation:

$$Score(d_i, p_j) = \sum_{w \in d_i} P(w|d_i) \times \log \frac{P(w|p_j)}{P(w|C)} \quad (1)$$

where C is the background model estimated on the entire collection. Smoothing will be employed to avoid the zero probability of unseen words. The timestamp of the document is the time partition maximizing a score according to the equation above.

In order to build temporal language models, a temporal corpus is needed. The temporal corpus can be any document collection where 1) the documents are timestamped with creation time, 2) covering a certain time period (at least the period of the queries collections), and 3) containing enough documents to make robust models. A good basis for such a corpus is a news archive. We will use the *New York Times annotated corpus*[1] since it is readily available for research purposes. However, any corpus with similar characteristics can be employed, including non-English corpora for performing dating of non-English texts. We will in the following denote a temporal corpus as $\mathcal{D}_{\mathcal{N}}$.

[1] `http://www.ldc.upenn.edu/Catalog/docs/LDC2008T19/new_york_times_annotated_corpus.pdf`

Table 1. Example of contents of temporal language models

Time	Term	Frequency
2001	World Trade Center	1545
2002	Terrorism	2236
2003	Iraq	1510
2004	Euro 2004	750
2004	Athens	1213
2005	Terrorism	1990
2005	Tsunami	3528
2005	Hurricane Katrina	1012
2008	Obama	2030

4 Determining Time of Queries using Temporal Language Models

In this section, we describe three approaches to determining the time of queries when no temporal criteria are provided. The first two approaches use temporal language models (cf. Sect. 3) as basis, and the last approach uses no language models. The first approach performs dating queries using keywords only. The second approach takes into account the fact that in general queries are short, and aims at solving this problem with a technique inspired by pseudo-relevance feedback (PRF) that uses the *top-k* retrieved documents in dating queries. The third approach also uses the *top-k* retrieved documents by PRF and assumes their creation dates as the time of queries.

All approaches will return a set of determined time intervals and their weights, which will be used in re-ranking documents in order to improve the retrieval effectiveness as described in more detail in Sect. 5.

4.1 Dating Query using Keywords

Our basic technique for query dating is based on using keywords only, and it is described formally in Algorithm 1.

The first step is to build temporal language models T_{LM} from the temporal document corpus (line 5), which essentially is the statistics of word usage (raw frequencies) in all time intervals, which are partitioned wrt. the selected time granularity g. Table 1 illustrates a subset of temporal language models. Creating the temporal language models (basically aggregating statistics grouped on time periods) is obviously a costly process, and will be done just once as an off-line process and then only the statistics have to be retrieved at query time.

For each time partition p_j in T_{LM}, the similarity score between q_{word} and p_j is computed (line 7). The similarity score is calculated using a normalized log-likelihood ratio according to Equation 1. Each time partition p_j and its computed score will be stored in C, or the set of time intervals and scores (line 8). After computing the scores for all time partitions, the contents of C will be sorted by similarity score, and then the *top-m* time intervals are selected as the output set A (line 10).

Finally, the determined time intervals resulting from Algorithm 1 will be assigned weights indicating their importance. In our approach, we simply give a weight to each time interval using its reverse ranked number. For example, if the output set A contains top-5 ranked time intervals, the intervals ranked 1, 2, 3, 4, and 5 will have the weights 5, 4, 3, 2, and 1 respectively.

Algorithm 1 *DateQueryKeywords*($q_{word}, g, m, \mathcal{D}_{\mathcal{N}}$)

1: **INPUT:** Query q_{word}, time granularity g, number of time intervals m, and temporal corpus $\mathcal{D}_{\mathcal{N}}$
2: **OUTPUT:** Set of time intervals associated to q_{word}
3: $A \leftarrow \emptyset$ *// Set of time intervals*
4: $C \leftarrow \emptyset$ *// Set of time intervals and scores*
5: $T_{LM} \leftarrow$ *BuildTemporalLM*$(g, \mathcal{D}_{\mathcal{N}})$
6: **for each** $\{p_j \in T_{LM}\}$ **do**
7: $score_{p_j} \leftarrow$ *CalSimScore*(q_{word}, p_j) *// Compute similarity score of q_{word} and p_j*
8: $C \leftarrow C \cup \{(p_j, score_{p_j})\}$ *// Store p_j and its similarity score*
9: **end for**
10: $A \leftarrow C$*.selectTopMIntervals(m)* *// Select top-m intervals ranked by scores*
11: **return** A

4.2 Dating a Query using Top-k Documents

In our second approach to query dating, the idea is that instead of dating query keywords q_{word} directly, we will instead date the *top-k* retrieved documents of the (non-temporal) query q_{word}. The resulting time of the query will be the combination of determined times of each top-k document.

The algorithm for dating a query using top-k retrieved documents is given in Algorithm 2. First, we retrieve documents by issuing a (non-temporal) query q_{word}, and retrieve only the *top-k* result documents (line 5). Then, temporal language models T_{LM} are built as described previously (line 6). For each document d_i in D_{TopK}, compute its similarity score with each time partition p_j in T_{LM} (lines 10-13). After computing scores for d_i for all time partitions, sort the contents of C by similarity score, and select only *top-m* time intervals as the results of d_i (line 14).

The next step is to update the set B with a set of time results C_{tmp} obtained from dating d_i. This is performed as follows: For each time interval p_k in C_{tmp}, check if B already contains p_k (line 16). If p_k exists in B, get a frequency of p_k and increase the frequency by 1 (lines 17-18). If p_k does not exist in B, add p_k into B as a new time interval and set its frequency to 1 (line 20). After dating all documents in D_{TopK}, sort the contents of B by frequency, and select only the *top-m* time intervals as the output set A (line 25).

The weights of time intervals will be their reverse ranked number. Note that it can be only one time interval in each rank of an output obtained from Algorithm 1, while it can be more than one time interval in each rank in case of Algorithm 2.

4.3 Using Timestamp of Top-k Documents

The last approach is a variant of the dating using *top-k* documents described above. The idea is similar in the use of the *top-k* retrieved documents of the (non-temporal) query q_{word}. The resulting time of the query will be the creation date (or timestamps) of each top-k document. In this case, no temporal language models are used.

Algorithm 2 *DateQueryWithTopkDoc*($q_{word}, g, m, k, \mathcal{D}_{\mathcal{N}}$)

INPUT: Query q_{word} , time granularity g, number of intervals and documents m, k, temporal corpus $\mathcal{D}_{\mathcal{N}}$
OUTPUT: Set of time intervals associated to q_{word}
$A \leftarrow \emptyset$ *// Set of time intervals*
$B \leftarrow \emptyset$ *// Set of time intervals and their frequencies*
$D_{TopK} \leftarrow$ *RetrieveTopKDoc*(q_{word}, k) *// Retrieve top-k documents*
$T_{LM} \leftarrow$ *BuildTemporalLM*$(g, \mathcal{D}_{\mathcal{N}})$
for each $\{d_i \in D_{TopK}\}$ **do**
  $C \leftarrow \emptyset$ *// Set of time intervals and scores*
  $C_{tmp} \leftarrow \emptyset$ *// Set of time intervals*
  for each $\{p_j \in T_{LM}\}$ **do**
    $score_{p_j} \leftarrow$ *CalSimScore*(d_i, p_j) *// Compute similarity score of* d_i *and* p_j
    $C \leftarrow C \cup \{(p_j, score_{p_j})\}$ *// Store* p_j *and its similarity score*
  end for
  $C_{tmp} \leftarrow C$*.selectTopMIntervals(m)* *// Select top-m intervals by scores*
  for each $\{p_k \in C_{tmp}\}$ **do**
    if B *has* p_k **then**
      freq $\leftarrow B$*.getFreqForTInterval*(p_k) *// Get frequency of* p_k
      $B \leftarrow B$*.updateFreqForTInterval*$(p_k, freq + 1)$ *// Increase frequency by 1*
    else
      $B \leftarrow B$*.addTInterval*$(p_k, 1)$ *// Add a new time interval and set its frequency to 1*
    end if
  end for
end for
$A \leftarrow B$*.selectTopMIntervals(m)* *// Select top-m intervals ranked by frequency*
return A

5 Re-ranking Documents using the Determined Time of Queries

In this section, we will describe how to use the time of queries determined by our approaches to improve the retrieval effectiveness. The idea is that, in addition to the documents' scores wrt. keywords, we will also take into account the documents' scores wrt. the *implicit* time of queries. Intuitively, documents with creation dates that closely match with the time of queries are more relevant and should be ranked higher.

There are a number of methods to combine a time score with existing text-based weighting models. For example, a time score can be combined with TF-IDF weighting using a linear combination, or it can be integrated into language modeling using a document prior probability as in [8]. In this paper, we propose to use a mixture model of a keyword score and a time score. Given a temporal query q with the determined time q_{time}, the score of a document d can be computed as follows:

$$S(q, d) = (1 - \alpha) \cdot S'(q_{word}, d_{word}) + \alpha \cdot S''(q_{time}, d_{time}) \quad (2)$$

where α is a parameter underlining the importance of a keyword score $S'(q_{word}, d_{word})$ and a time score $S''(q_{time}, d_{time})$. A keyword score $S'(q_{word}, d_{word})$ can be implemented using any of existing text-based weighting models, and it can be normalized as

$S'_{norm}(q_{word}, d_{word}) = \frac{S'(q_{word}, d_{word})}{\max S'(q_{word}, d_{word,i})}$ where $\max S'(q_{word}, d_{word,i})$ is the maximum keyword score among all documents.

For a time score $S''(q_{time}, d_{time})$, we formulate the probability of generating the time of query q_{time} given the associated time partition of document d_{time} as:

$$\begin{aligned} S''(q_{time}, d_{time}) &= P(q_{time}|d_{time}) \\ &= P(\{t'_1, \ldots, t'_n\}\,|d_{time}) \\ &= \frac{1}{|q_{time}|} \sum_{t'_j \in q_{time}} P(t'_j|d_{time}) \end{aligned} \tag{3}$$

where q_{time} is a set of time intervals $\{t'_1, \ldots, t'_n\}$ and $(t'_1 \cap t'_2 \cap \ldots \cap t'_n) = \emptyset$. So, $P(q_{time}|d_{time})$ is an average of the probability of generating a time interval, or $P(t'_j|d_{time})$, over all the number of time intervals in q_{time}, or $|q_{time}|$.

The probability of generating a time interval t'_j given the time partition of document d_{time} can be defined in two ways as proposed in [1]: 1) ignoring uncertainty, and 2) taking uncertainty into account. By ignoring uncertainty, $P(t'_j|d_{time})$ is defined as:

$$P(t'_j|d_{time}) = \begin{cases} 0 & \text{if } d_{time} \neq t'_j, \\ 1 & \text{if } d_{time} = t'_j. \end{cases} \tag{4}$$

In this case, the probability of generating query time will be equal to 1 only if d_{time} is exactly the same as t'_j. By taking into account a weight of each time interval t'_j, $P(t'_j|d_{time})$ with *uncertainty-ignorant* becomes

$$P(t'_j|d_{time}) = \begin{cases} 0 & \text{if } d_{time} \neq t'_j, \\ \frac{w(t'_j)}{\sum_{t'_k \in q_{time}} w(t'_k)} & \text{if } d_{time} = t'_j. \end{cases} \tag{5}$$

where $w(t'_j)$ is a function giving a weight for a time interval t'_j, which is normalized by the sum of all weights $\sum_{t'_k \in q_{time}} w(t'_k)$.

In the case where uncertainty is concerned, $P(t'_j|d_{time})$ is defined using an exponential decay function:

$$P(t'_j|d_{time}) = DecayRate^{\lambda \cdot |t'_j - d_{time}|} \tag{6}$$

where $DecayRate$ and λ are constant, $0 < DecayRate < 1$ and $\lambda > 0$. Intuitively, this function gives a probability that decreases proportional to the difference between a time interval t'_j and the time partition of document d_{time}. A document with its time partition closer to t'_j will receive a higher probability than a document with its time partition farther from t'_j. By incorporating a weight of each time interval t'_j, $P(t'_j|d_{time})$ with *uncertainty-aware* becomes

$$P(t'_j|d_{time}) = \frac{w(t'_j)}{\sum_{t'_k \in q_{time}} w(t'_k)} \times DecayRate^{\lambda \cdot |t'_j - d_{time}|} \tag{7}$$

The normalization of $S''_{norm}(q_{time}, d_{time})$ can be computed in two ways: 1) uncertainty-ignorant using $P(t'_j|d_{time})$ defined in Equation 5, and 2) uncertainty-aware using $P(t'_j|d_{time})$

defined in Equation 7. Finally, the normalized value of $S''_{norm}(q_{time}, d_{time})$ will be substituted $S''(q_{time}, d_{time})$ in Equation 8 yielding the normalized score of a document d given a temporal query q with determined time q_{time} as follows:

$$S_{norm}(q, d) = (1 - \alpha) \cdot S'_{norm}(q_{word}, d_{word}) + \alpha \cdot S''_{norm}(q_{time}, d_{time}) \quad (8)$$

6 Experiments

In this section, we will perform two experiments in order to evaluate our proposed approaches: 1) determining the time of queries using temporal language models, and 2) re-ranking search results using the determined time. In this section, we will describe the setting for each of the experiments, and then the results.

6.1 Experimental Setting

As we mentioned earlier, we can use any news archive collection to create temporal language models. In this paper, we used the New York Times annotated corpus as the temporal corpus. This collection contains over 1.8 million articles covering a period of January 1987 to June 2007. The temporal language models were created and stored in databases using Oracle Berkeley DB version 4.7.25.

To evaluate the query dating approaches, we obtained queries from Robust2004, which is a standard test collection for the TREC Robust Track containing 250 topics (topics 301-450 and topics 601-700). As reported in [8], some TREC queries favor documents in particular time periods. Similarly, we analyzed a distribution of relevant documents of the Robust2004 queries over time, and we randomly selected 30 strongly time-related queries (with the topic number: 302, 306, 315, 321, 324, 330, 335, 337, 340, 352, 355, 357, 404, 415, 428, 435, 439, 446, 450, 628, 648, 649, 652, 653, 656, 667, 670, 676, 683, 695). Time intervals of relevant documents were assumed as the correct time of queries. We measured the performance using precision, recall and F-score. Precision is the fraction of determined time intervals that are correct, while recall indicates the fraction of correct time intervals that are determined. F-score is the weighted harmonic mean of precision and recall, where we set $\beta = 2$ in order to emphasize recall. For query dating parameters, we used the top-m interval with $m = 5$, and the time granularity g and the *top-k* documents were variable in the experiments.

To evaluate the re-ranking approaches, the Terrier search engine was employed, and we used the BM25 probabilistic model with Generic Divergence From Randomness (DFR) weighting as our retrieval model. For the simplicity, we used default parameter settings for the weighting function. Terrier provides a mechanism to alter scores for retrieved documents by giving prior scores to the documents. In this way, we re-ranked search results at the end of retrieval by combining a keyword score $S'(q_{word}, d_{word})$ and a time score $S''(q_{time}, d_{time})$ as defined in Equation 8. We conducted re-ranking experiments using two collections: 1) the Robust2004 collection, and 2) the New York Times annotated corpus. For the Robust2004 collection, we used the 30 queries as temporal queries without time explicitly provided. The retrieval effectiveness of temporal search using the Robust2004 collection is measured by Mean Average Precision (MAP), and

Table 2. Example of the Google zeitgeist queries and associated time intervals

Query	Time	Query	Time
diana car crash	1997	madrid bombing	2005
world trade center	2001	pope john paul ii	2005
osama bin laden	2001	tsunami	2005
london congestion charges	2003	germany soccer world cup	2006
john kerry	2004	torino games	2006
tsa guidelines liquids	2004	subprime crisis	2007
athens olympics games	2004	obama presidential campaign	2008

R-precision. For the New York Times annotated corpus, we selected 24 queries from a historical collection of aggregated search queries, or the Google zeitgeist[2]. An example of temporal queries are shown in Table 2. The temporal searches were conducted by human judgment. Performance measures are the precision at 5, 10, and 15 documents, or P@5, P@10, and P@15 respectively. For re-ranking parameters, we used an exponential decay rate $DecayRate = 0.5$, and $\lambda = 0.5$. A mixture model parameter was obtained from the experiments, where $\alpha = 0.05$ and 0.10 for *uncertainty-ignorant* and *uncertainty-aware* methods respectively.

The description of different approaches is given as follows. **QW** determines time using keywords *plus* uncertainty-ignorant re-ranking. **QW-U** determines time using keywords *plus* uncertainty-aware re-ranking. **PRF** determines time using top-k retrieved documents *plus* uncertainty-ignorant re-ranking. **PRF-U** determines time using top-k retrieved documents *plus* uncertainty-aware re-ranking. **NLM** assumes creation dates of top-k retrieved documents as the time of queries (no language models used) *plus* uncertainty-ignorant re-ranking. **NLM-U** assumes creation dates of top-k retrieved documents as the time of queries (no language models used) *plus* uncertainty-aware re-ranking. Top-k documents were retrieved using pseudo relevance feedback, i.e., the result documents after performing query expansion using Rocchio algorithm.

6.2 Experimental Results

The performance of query dating methods are shown in Table 3. NLM performs best in precision for all time granularities whereas PRF performs best in recall (only for *12-month*). NLM and PRF give the best F-score results for *6-month* and *12-month* respectively. In general, the smaller k tends to give the better results, while *12-month* yields higher performance compared to *6-month*. Finally, the performance of QW seems to be robust for *12-month* regardless of dating solely short keywords.

To evaluate re-ranking, the baseline of our experiments is a retrieval model without taking into account the time of queries, i.e., pseudo relevance feedback using Rocchio algorithm. For the Robust2004 queries, the baseline performance are MAP=0.3568 and R-precision=0.3909. Experimental results of MAP and R-precision are shown in Table 4. The results show that QW,QW-U,PRF,PRF-U outperformed the baseline in both MAP and R-precision for *12-month*, and NLM,NLM-U outperformed the baseline in all cases. PRF-U always performed better than PRF in both MAP and R-precision for *12-month*, while QW-U performed better than QW in R-precision for *12-month* only.

[2] `http://www.google.com/intl/en/press/zeitgeist/index.html`

Table 3. Query dating performance using precision, recall and F-score

Method	Precision		Recall		F-score($\beta = 2$)	
	6-month	*12-month*	*6-month*	*12-month*	*6-month*	*12-month*
QW	.56	.67	.34	.64	.37	.65
PRF *(k=5)*	.55	.63	.47	**.79**	.48	**.75**
PRF *(k=10)*	.56	.60	.46	.74	.48	.71
PRF *(k=15)*	.54	.60	.42	.70	.44	.68
NLM *(k=5)*	.92	**.97**	.35	.44	.40	.49
NLM *(k=10)*	.90	.95	.48	.56	.53	.61
NLM *(k=15)*	.89	.93	.56	.63	.61	.67

Table 4. Re-ranking performance using MAP and R-precision with the baseline performance 0.3568 and 0.3909 respectively (the Robust2004 collection)

Method	MAP		R-precision	
	6-month	*12-month*	*6-month*	*12-month*
QW	.3565	.3576	.3897	.3924
QW-U	.3556	.3573	.3925	.3943
PRF *(k=5)*	.3564	.3570	.3885	.3926
PRF *(k=10)*	.3568	.3570	.3913	.3919
PRF *(k=15)*	.3566	.3567	.3912	.3921
PRF-U *(k=5)*	.3548	.3574	.3903	.3950
PRF-U *(k=10)*	.3538	.3576	.3904	.3935
PRF-U *(k=15)*	.3538	.3572	.3893	.3940
NLM *(k=5)*	.3585	.3589	.3924	.3917
NLM *(k=10)*	.3586	.3591	.3918	.3925
NLM *(k=15)*	.3584	.3596	.3898	.3934
NLM-U *(k=5)*	.3604	.3608	.3975	**.3978**
NLM-U *(k=10)*	.3604	.3610	.3953	.3961
NLM-U *(k=15)*	.3606	**.3620**	.3943	.3967

NLM, NLM-U always outperformed the baseline and the other proposed approaches because using the creation dates of documents is more accurate than those obtained from the dating process. This depicts that taking time into re-ranking can better the retrieval effectiveness. Hence, if query dating is improved with a high accuracy, the retrieval effectiveness will be improved significantly.

The results of evaluate the Google zeitgeist queries are shows in Table 5. In this case, we fix the number of *top-k* to 15 only. Table 5 illustrated the precision at 5, 10 and 15 documents. The baseline performance is P@5=0.35, P@10=0.30 and P@15=0.27. The results show that our proposed approaches perform better than the baseline in all cases. NLM, NLM-U performs the best among all proposed approaches.

7 Conclusions and Future Work

In this paper, we have studied implicit temporal queries where no temporal criteria is provided, and how to increase retrieval effectiveness for such queries. The effectiveness

Table 5. Re-ranking performance using P@5, P@10, and P@15 with the baseline performance 0.35, 0.30 and 0.27 respectively * indicates statistically improvement over the baselines using t-test with significant at $p < 0.05$ (the NYT collection)

Method	P@5		P@10		P@15	
	6-month	*12-month*	*6-month*	*12-month*	*6-month*	*12-month*
QW	.42	.45	.37	.39	.32	.33
QW-U	.40	.42	.35	.36	.30	.32
PRF *(k=15)*	.42	.46	.38	.42	.35	.39
PRF-U *(k=15)*	.41	.45	.36	.40	.33	.37
NLM *(k=15)*	.50	.52	.47	.49	.42	.44
NLM-U *(k=15)*	.53	**.55***	.48	**.50***	.45	**.46***

has been improved by determining the implicit time of the queries and employing this to re-rank the query results. Through extensive experiments we show that our proposed approach improves retrieval effectiveness.

Although using our approach shows improvement on retrieval effectiveness, the quality of the actual query dating processing is a limitation when aiming at further increase in effectiveness. Future work includes further improvement on the query dating based on external knowledge from sources like Wikipedia.

References

1. K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *Proceedings of ECIR'2010*, 2010.
2. K. Berberich, S. J. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *Proceedings of SIGIR'2007*, 2007.
3. F. de Jong, H. Rode, and D. Hiemstra. Temporal language models for the disclosure of historical text. In *Proceedings of AHC'2005 (History and Computing)*, 2005.
4. F. Diaz and R. Jones. Using temporal profiles of queries for precision prediction. In *Proceedings of the 27th SIGIR*, 2004.
5. A. Jatowt, Y. Kawai, and K. Tanaka. Temporal ranking of search engine results. In *Proceedings of WISE*, 2005.
6. N. Kanhabua and K. Nørvåg. Improving temporal language models for determining time of non-timestamped documents. In *Proceedings of ECDL'2008*, 2008.
7. W. Kraaij. Variations on language modeling for information retrieval. *SIGIR Forum*, 39(1):61, 2005.
8. X. Li and W. B. Croft. Time-based language models. In *Proceedings of CIKM*, 2003.
9. D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proceedings of SIGIR'2009*, 2009.
10. K. Nørvåg. Supporting temporal text-containment queries in temporal document databases. *Journal of Data & Knowledge Engineering*, 49(1):105–125, 2004.
11. S. Nunes, C. Ribeiro, and G. David. Use of temporal expressions in web search. In *Proceedings of ECIR'2008*, 2008.
12. J. Perkiö, W. Buntine, and H. Tirri. A temporally adaptive content-based relevance ranking algorithm. In *Proceedings of the 28th SIGIR*, 2005.
13. N. Sato, M. Uehara, and Y. Sakai. Temporal ranking for fresh information retrieval. In *Proceedings of the 6th IRAL*, 2003.