Answering General Time-Sensitive Queries

Wisam Dakka, Luis Gravano, and Panagiotis G. Ipeirotis, *Member*, *IEEE*

Abstract—Time is an important dimension of relevance for a large number of searches, such as over blogs and news archives. So far, research on searching over such collections has largely focused on locating topically similar documents for a query. Unfortunately, topic similarity alone is not always sufficient for document ranking. In this paper, we observe that, for an important class of queries that we call *time-sensitive queries*, the publication time of the documents in a news archive is important and should be considered in conjunction with the topic similarity to derive the final document ranking. Earlier work has focused on improving retrieval for "recency" queries that target recent documents. We propose a more general framework for handling time-sensitive queries and we automatically identify the important time intervals that are likely to be of interest for a query. Then, we build scoring techniques that seamlessly integrate the temporal aspect into the overall ranking mechanism. We present an extensive experimental evaluation using a variety of news article data sets, including TREC data as well as real web data analyzed using the Amazon Mechanical Turk. We examine several techniques for detecting the important time intervals for a query over a news archive and for incorporating this information in the retrieval process. We show that our techniques are robust and significantly improve result quality for time-sensitive queries compared to state-of-the-art retrieval techniques.

Index Terms—Information search and retrieval, processing time-sensitive queries, time-sensitive search.

1 Introduction

Time is an important dimension of relevance for a large number of searches, such as over blogs and news archives. So far, research on searching over such collections has largely focused on retrieving topically similar documents for a query. Unfortunately, ignoring or not fully exploiting the time dimension can be detrimental for a large family of queries for which we should consider not only the document topical relevance but the publication time of the documents as well, as demonstrated by the following example:

Example 1. *Consider the query* [Madrid bombing] *over the news archive of a state-of-the-art multidocument summarization system that crawls and summarizes news articles from the web on a daily basis. Fig. 1 zooms in on a portion of the histogram for the query results, reporting the number of matching documents in the news archive for each day between January and December 2004. This histogram reveals particular time intervals that are likely to be of special interest for the query, such as the month of March 2004, when a terrorist group bombed trains in Madrid. The same figure shows an analogous histogram for query* [Google IPO]*: the "peaks" in the histogram coincide with two important events, namely, the announcement of the Google IPO and, a few months later, the actual IPO.*

- *W. Dakka is with Google, 600 Amphitheatre Parkway, Mountain View, CA 94043. E-mail: wisam@google.com.*
- *L. Gravano is with the Computer Science Department, Columbia University, 1214 Amsterdam Avenue, New York, NY 10027. E-mail: gravano@cs.columbia.edu.*
- *P.G. Ipeirotis is with the Information Systems Group, IOMS Department, Leonard N. Stern School of Business, New York University, KMC 8-84, 44 West 4th Street, New York, NY 10012. E-mail: panos@stern.nyu.edu.*

Manuscript received 28 Mar. 2009; revised 6 Jan. 2010; accepted 5 July 2010; published online 27 Sept. 2010.
Recommended for acceptance by B.C. Ooi.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-03-0214.
Digital Object Identifier no. 10.1109/TKDE.2010.187.

These examples motivate two observations on searching over news archives. First, topic-similarity ranking does not model time explicitly, which means that the important dimension of time is not considered directly when deciding on the results that are returned for a user query. The various "peaks" in the Fig. 1 histograms, which reveal important information for the queries, are thus not leveraged to produce high-quality query results. Second, a topic-similarity ranking of the query results often does not reflect the distribution of relevant documents over time. In fact, for many queries, users have a general—but often vague and unspecified—idea about the relevant time periods for the queries. For example, the query *[Madrid bombing]* might (implicitly) be after articles from March and April 2004. So perhaps a better formulation of the query would be *[Madrid bombing* ***prefer****: 03/11/2004-04/30/2004]*, indicating the relevant time interval for the event.

In this paper, we observe that, for an important class of queries over news archives that we call *time-sensitive queries*, topic similarity is not sufficient for ranking. For such queries, the publication time of the documents is important and should be considered in conjunction with the topic similarity to derive the final document ranking. Most current methods for searching over large archives of timed documents incorporate time in a relatively crude manner: users can submit a keyword query, say *[Madrid bombing]*, and restrict the results to articles written between March and April 2004, or alternatively sort the results on the publication date of the articles. Unfortunately, searchers do not always know the appropriate time intervals for their queries, and placing the burden on the users to explicitly handle time during querying is not desirable.

Toward helping users understand the temporal distribution of matching articles for a query, Google's News Archive Search[1] supplements query results with a "timeline,"

1. http://news.google.com/archivesearch.

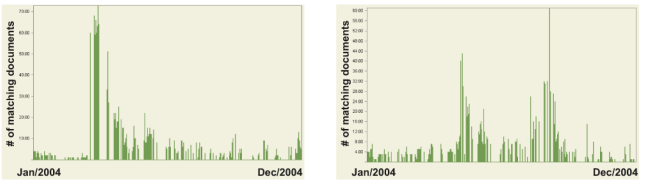

Fig. 1. Histograms for queries *[Madrid bombing]* and *[Google IPO]*, showing the number of documents with all query words for each day from January to December 2004 in a news archive.

showing the distribution of query matches over time. Google also highlights key time periods for each query (e.g., the year 2004 is marked for the query *[Madrid bombing]*), so users can explicitly restrict the search to a specific time period. Similarly, Jones and Diaz [1] show how to exploit query result timelines to decide whether to ask users to select appropriate time periods for their queries. In Section 3, we discuss several techniques to estimate the temporal relevance of a day to a query at hand. These estimation techniques use the temporal distribution of matching articles for the query to compute the probability that a day in the archive has a relevant document for the query.

Beyond asking for explicit user input, earlier work by Li and Croft [2] focused on handling *recency queries*, which are queries that are after recent events or breaking news. Examples of recency queries are *[NYC crane collapse]* in May 2008, or *[Sarkozy French elections]* in April 2007. Li and Croft's time-sensitive approach processes a recency query by computing traditional topic-similarity scores for each document, and then "boosts" the scores of the most recent documents, to privilege recent articles over older ones. In contrast to traditional models, which assume a uniform prior probability of relevance $p(d)$ for each document d in a collection, Li and Croft define the prior $p(d)$ to be a function of document d's creation date. The prior probability $p(d)$ decreases exponentially with time, and hence recent documents are ranked higher than older documents. Li and Croft's strategy is designed for queries that are after recent documents, but it does not handle other types of time-sensitive queries, such as *[Madrid bombing]*, *[Google IPO]*, or even *[Sarkozy French elections]* (in May 2008), that implicitly target one or more past time periods.

In this paper, we propose a more general framework for answering time-sensitive queries that builds on and substantially expands the earlier work on recency queries. If the relevant time period for a time-sensitive query is unspecified, several query processing approaches are possible. One alternative is to automatically suggest, based on the query terms, relevant time ranges for the query and allow users to explicitly select appropriate time intervals [1]. As an alternative that demands less input from the users, and which we follow in this paper, we can automate the previous procedure and prioritize results from periods that we automatically identify as relevant. We can then naturally define the relevance of a document as a combination of topic similarity and time relevance.

Specifically, in this paper, we propose a general framework to incorporate time into the retrieval task in a principled manner. For a given time-sensitive query over a news archive, our approach automatically identifies important time intervals for the query. These intervals are then used to adjust the document relevance scores by boosting the scores of documents published within the important intervals. We have implemented our system on top of Indri,[2] a state-of-the-art search engine that combines language models and inference networks for retrieval [3], as well as over Lemur,[3] into its implementation of BM25 [4], [5], [6], [7]. Our system provides a web interface for searching the Newsblaster archive,[4] an operational news archive and summarization system, and for experimenting with variations of our approach. We present an extensive evaluation of our system, using both TREC data and real web data analyzed using the Amazon Mechanical Turk.[5] Our experiments show that the quality of the results produced by our techniques for time-sensitive queries is significantly higher than that of the (strong) baselines that we consider.

The rest of the paper is organized as follows: Section 2 discusses a general class of queries that we call time-sensitive queries. In Section 3, we introduce the notion of temporal relevance, which is the probability of a document published at a certain time to be relevant to a given query, and we also present three techniques for computing this probability. In Section 4, we integrate temporal relevance with state-of-the-art retrieval models, including a query likelihood (QL) model, a relevance model (RM), a probabilistic relevance model (PRM), and a query expansion with pseudorelevance feedback model, to naturally process time-sensitive queries. In these models, we combine topical relevance and temporal relevance to determine the overall relevance of a document. Then, Section 5 reports the experimental settings and results. Finally, Sections 6 and 7 describe related work and conclude the paper. A short description of a preliminary version of this work appeared in [8].

2 Time-Sensitive Queries

For recency queries [2], the bulk of the relevant documents is, by definition, from recent days. For other families of queries, the relevant documents may be distributed differently over the time span of a news archive. For example, the query *[Madrid bombing]* (Fig. 1) executed on a news archive might be after articles about the specific details of the Madrid train bombing at the time it happened, so this query might be considered a *past* query. More generally, relevant results for some queries may exist in certain time periods, in which sudden, large-scale news coverage relevant to the queries takes place and diminishes after a period of time. Other queries, such as *[Barack Obama]*, are likely to be after relevant results from multiple "events." These queries are examples of *time-sensitive* queries, which we define as follows:

Definition 1. *A query over an archive of time stamped news documents is 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. http://www.lemurproject.org/indri.
3. http://www.lemurproject.org.
4. Due to copyright-related issues, Newsblaster is password protected.
5. http://www.mturk.com.

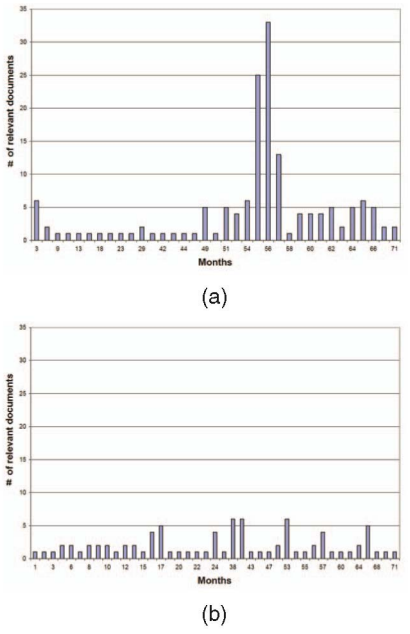

Fig. 2. Relevant-document histograms of a time-sensitive (a) and a time-insensitive (b) query from a TREC ad hoc query set. (a) Query #311, *[Industrial Espionage]*, a time-sensitive query. (b) Query #304, *[Endangered Species (Mammals)]*, a time- insensitive query.

To illustrate the difference between time-sensitive and time-insensitive queries, Fig. 2 shows a histogram of both a time-sensitive query (TREC query number 311) and a time-insensitive query (TREC query number 304), from the TREC ad hoc title queries 301-350. Unlike the histograms we showed in Fig. 1, Fig. 2 shows the true distribution of the relevant documents, not of the matching documents. Fig. 3 shows the histograms of the number of matching documents for several real-life, time-sensitive queries.

News archives often include many matching documents for time-sensitive queries. For example, the query *[Saddam Hussein capture]* has 936 matching stories in The New York Times archive, as of March 2009. We claim that traditional topic-similarity ranking alone may not be desirable for time-sensitive queries, where we can explicitly account for time to produce high-quality query results. *Our basic intuition is that the relevance of one document for a given query provides us with useful information about the relevancy of other documents with similar content that were published around the same time.* This is in contrast to "traditional" information retrieval engines, which consider the relevancy of each document in isolation. In the next section, we discuss our first step in accounting for time by introducing techniques to estimate temporal relevance, which is the probability that a time period is relevant to a query at hand.

3 Temporal Relevance

Typically, documents in archival collections are stamped with their publication dates. Unfortunately, queries often are answered and ranked without consideration of these time stamps, with the exception of some user feedback to sort by date or restrict answers to a time range. To answer the type of time-sensitive queries discussed in Section 2 over a news archive, we would like to use the temporal information

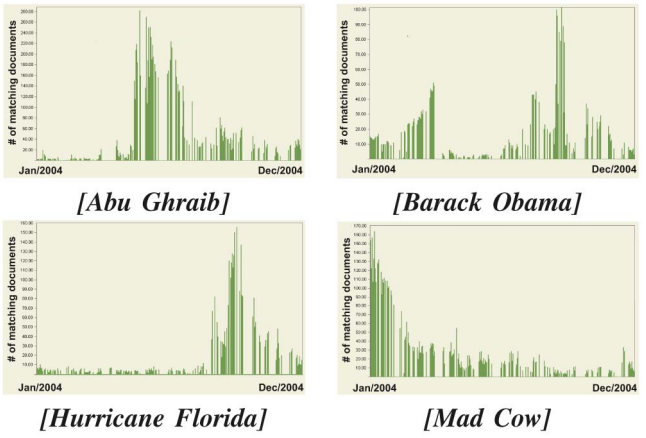

Fig. 3. Histograms for several queries, revealing different distributions of the query terms over the Newsblaster archive from January to December 2004.

implicitly available in the archive. For this, we observe that time-sensitive queries are generally after documents from specific time periods. For example, the majority of documents relevant to the TREC query *[Industrial Espionage]* (Fig. 2) are located within a specific time period.

This observation suggests that it is important to know the distribution of relevant documents over time for a given query. We hypothesize that this distribution can be used to improve the answer quality for a retrieval task (see Section 4). Therefore, we could attempt to compute the probability $p(t|q)$ that a day t is relevant to a query q using the distribution of relevant documents, and use this value for answering the query q. Unfortunately, we usually have no a priori knowledge of the relevant documents for a given query and, as a result, we cannot accurately compute this distribution. In this section, we investigate several ways to estimate $p(t|q)$ based on statistics readily extractable from a news archive.

3.1 Estimation Using "Ground Truth"

We first discuss a hypothetical situation in which we know, for a given query q, its complete set of relevant documents R_q, which we refer to as the "ground truth" for the query. In this situation, we can estimate $p(t|q)$ based solely on R_q. Specifically, consider a news archive D. Then, according to Bayes' rule, $p(t|q)$ is

$$p(t|q) = \frac{p(q|t) \cdot p(t)}{p(q)} = \frac{p(q|t) \cdot p(t)}{\sum_{\hat{t} \in dates(D)} p(q|\hat{t}) \cdot p(\hat{t})}, \qquad (1)$$

where $dates(D)$ is the time span of D; $p(t)$ is the probability that day t contains a document (relevant to q or not), multinomially distributed over t; $p(q|t)$ follows a Bernoulli distribution and is the probability that the documents published in t are relevant for answering query q (e.g., a random news document selected among the documents published in April 2008, when Barack Obama lost the Democratic primaries in Pennsylvania, has higher chances of being relevant to the query *[Obama Pennsylvania]*, compared to a random news document selected among the documents published in April 2007); and $p(q)$ is the prior probability of finding a document relevant to q, and serves as a normalizing factor.

Now, since we (assume that we) know the complete set of documents R_q that are relevant to q, we can directly estimate $p(q|t)$ using the distribution of the documents in R_q over time as follows:

$$p(q|t) = \frac{count(R_q, t)}{count(D, t)}, \tag{2}$$

where $count(R_q, t)$ is the number of (relevant) documents in R_q that were published at time t and $count(D, t)$ is the number of documents that were published at time t in the collection D. Since $p(t) = \frac{count(D,t)}{|D|}$, combining (2) with (1) results in

$$p(t|q) = \frac{count(R_q, t)}{|R_q|}, \tag{3}$$

where the relevance of the day t is then estimated as the fraction of the relevant documents published at time t.

Unfortunately, we generally do not know the ground truth R_q for a given query q, as mentioned above. Because of this, we now present three approaches to estimate $p(t|q)$ without knowing the set of relevant documents R_q.

3.2 Direct Estimation Using the Distribution of Matching Documents

To estimate $p(t|q)$ for a query q and time t in the absence of knowledge of the relevant documents for q and time t, Jones and Diaz [1] suggested using the top matching documents for q and their relevance scores, as provided by some underlying retrieval model. Specifically, $p(t|q)$ is defined as a normalized, weighted sum of the relevance scores of the top-k matching documents published at day t, as follows:

$$p(t|q) \approx \sum_{d \in D_{q,k}} p(t|d) \cdot \frac{p(d|q)}{\sum_{\hat{d} \in D_{q,k}} p(\hat{d}|q)}, \tag{4}$$

where $D_{q,k}$ are the top-k documents from D for query q, according to some underlying retrieval model, and $p(t|d) = 1$ if d's publication time is equal to t, otherwise $p(t|d) = 0$. To select the best k, Jones and Diaz train the model using a set of queries and their relevance judgments.

Based on this definition, most days are assigned a probability of zero for a query as the top-k documents for the query will tend to cover just a small portion of the days represented in the news archive. In addition, this definition does not account for the differing number of documents published each day and assumes the number to be constant. To rectify this, Jones and Diaz apply a background probability smoothing using the distribution of documents across collection D on (4) as follows:

$$\hat{p}(t|q) \approx \lambda \cdot p(t|q) + (1 - \lambda) \cdot \frac{1}{|D|} \sum_{d \in D} p(t|d), \tag{5}$$

$$\approx \lambda \cdot p(t|q) + (1 - \lambda) \cdot p(t). \tag{6}$$

To take into account the fact that news stories about a single topic or event may occur over a period of several days, a second step of smoothing is necessary. To connect the temporal relevance value of a day t with days in the near past, Jones and Diaz [1] apply simple moving-average smoothing, a technique that has been explored in the field of time series analysis, on (6) as follows:

$$\bar{p}(t|q) \approx \frac{1}{x} \cdot \sum_{i=0}^{x-1} \hat{p}(t - i|q), \tag{7}$$

where x is the number of days we consider for smoothing.

In the following section, we present techniques that estimate $p(t|q)$ in a different fashion, to be less dependent on the underlying retrieval model, to have a choice of several smoothing alternatives, as well as to have full control on the "shape" of the probability distribution.

3.3 Estimation Using Binning

The previous technique, from Jones and Diaz [1], relies heavily on the underlying retrieval model to estimate $p(t|q)$. The retrieval model not only suggests the top-k matching documents as an approximation to the true relevant documents, but also weights these documents based on their relevance scores. These scores are, in turn, used to determine the contribution of each top-k document to the final temporal relevance value of the document's publication day. This *direct* dependency on the relevance scores for estimating the $p(t|q)$ values is somewhat problematic, because these scores were designed for a different purpose, namely, document ranking. Furthermore, the previous technique is not conducive to exploring different "shapes" of the $p(t|q)$ probability distribution. Now, we suggest a general framework to estimate $p(t|q)$ that addresses these issues, so that it is less dependent on the underlying retrieval model by considering only the top-k matching documents without using their relevance scores directly.

In Section 3.1, we defined $p(t|q)$ to be proportional to $p(q|t) \cdot p(t)$. Since $p(t)$ can be either uniform, if all days in the archive have roughly the same number of documents, or directly computed as $\frac{count(D,t)}{|D|}$, we focus on estimating $p(q|t)$. Specifically, we explore defining $p(q|t)$ for a query q and time t by analyzing the number of documents matching the query q over time. Our conjecture is that certain patterns of matching frequencies over time might help identify time periods relevant to the query. For example, an abrupt change in frequency between consecutive days might indicate the presence of an event that is relevant to the query. To illustrate this observation, consider query *[Google IPO]*. The number of matching documents for this query was historically very low, but it increased by a large margin around the time when Google announced its IPO and also when it went public, as discussed above. In other words, we may use the number of matches found to estimate the probability of generating the query q from the documents published in a given time t.

Specifically, to estimate $p(q|t)$, we propose to arrange all time periods into *bins*, such that each bin represents a different "priority level." We then order these bins based on their priority and assign estimated relevance values to the time periods in these bins accordingly. For example, the query-frequency histogram of the query *[Barack Obama]* reveals relevant time intervals in the years 2002 and 2004 (and beyond); accordingly, these time periods will be assigned to a "high priority" bin. Note that Li and Croft's work [2] on recency queries can be easily mapped to this framework if we arrange the documents in bins according

to their creation date, so that recent documents are located in bins with the highest prior level.

Algorithm 1 describes our method to estimate the value $p(q|t)$ of each time t for a given query q over a news archive D, based on the publication time of the documents in D. We follow three basic steps: Step 1 generates a query-frequency histogram (Section 3.3.1); then, Step 2 analyzes the histogram and partitions the time periods into bins accordingly (Section 3.3.2); and finally, Step 3 estimates the $p(q|t)$ values based on the binning of the time periods (Section 3.3.3).

Algorithm 1. General time-based approach for estimating the value $p(q|t)$ of each time t and a query q.

Input: Query q, document collection D
Output: Time-based probability $p(q|t)$ for each time t
Step 1: Compute the query-frequency histogram for q using the publication time of the documents in D (Section 3.3.1).

Step 2: Partition the times into bins $b_0, \ldots, b_\ell$ based on the histogram characteristics (Section 3.3.2).

Step 3: Define the value $p(q|t)$ of each time t based on t's bin, such that a time in b_i will have a higher value than a time in b_j if $i < j$ (Section 3.3.3).

3.3.1 Generating Query-Frequency Histograms

As the first step of our approach (Algorithm 1), we produce a query-frequency histogram for a user query executed over a news archive by identifying all the documents in the archive that match the query. For certain queries, using conjunctive Boolean semantics for matching is sufficient to draw an expressive histogram that approximates the real distribution of relevant time periods. However, for most of the TREC queries with which we experimented (Section 5.1), the conjunctive Boolean model returns very few or no documents at all, so we resorted to the more flexible query likelihood model. Specifically, we use the query likelihood model [9], [10] to retrieve the top-k documents for query q with the highest $p(q|d)$ values.[6] Then, we use the publication time of the returned documents to generate the query-frequency histogram over time.

3.3.2 Analyzing Histograms

After generating the query-frequency histogram, we move to Step 2 of our approach (Algorithm 1) and analyze the histogram to organize all times (days) into bins. The first bin corresponds to the most "important" times for the query, according to the histogram, the second bin corresponds to the next level of importance, and so on. We explore alternate binning techniques based on different underlying hypotheses on how to identify the important time intervals.

Daily frequency (DAY). Consider query *[Goldman report earnings]*. This query is likely to be after documents discussing earnings reports for the Wall Street investment bank Goldman Sachs, which are published once per quarter and cause news coverage to spike briefly. Hence, the "events" that this query likely targets last one or two days and, as a result, the query-frequency histogram of such a query will usually have sharp, thin "spikes" indicating these events. Then, to characterize time-sensitive queries, we might attempt to isolate daily spikes and use them to estimate the $p(q|t)$ values. For this, we simply assign a day t to a bin based on the number of matching documents for the query that were published in the same day t. In the end, our documents are organized in bins $b_0, \ldots, b_\ell$, where bin b_0 will contain the day(s) with the highest number of published documents, b_1 will correspond to the day(s) with the second-largest frequency, and so on.

Fixed interval frequency (FIXED). News events can last for longer than one or two days. As a result, spikes across multiple consecutive days that correspond to such an event appear in the query-frequency histogram, creating the shape of a "bump." The query frequency during these periods is relatively higher than during other time periods. To determine if a particular day is part of a bump, we measure the average daily query frequency in each day of a time interval of a fixed size starting on that day. This is a modified version of the previous technique, where instead of studying each individual day in isolation, we consider the average daily frequency in a larger interval of x consecutive days. (We experimented with different values of x, as we will discuss in Section 5.3.) In the end, b_0 will contain all the days with the highest average frequency in their windows, b_1 will correspond to the second-largest average frequency, and so on.

Moving window (WIN). In this technique, we consider the query frequency in both past and future days to determine if a particular day is part of a bump. This technique is a generalization of the previous one. For each day, we compute the average daily query frequency in a window of x days into the past and x days into the future. (We experimented with different values of x, as we will discuss in Section 5.3.) In the end, bin b_0 will contain all the days with the highest average daily frequency in their windows, b_1 will correspond to the second-largest average frequency, and so on.

Running mean (MEAN). Phrases or words such as *war on terror*, *WMD*, *Shiite*, and *iPod* used to be infrequent. At one point, major events, such as the 11 September 2001 terror attack or the war on Iraq, occur and cause new terms and phrases to come into common usage. The above binning techniques assumed that if two days have the same query-frequency value (raw value or averaged value), then documents from both days should be placed in the same bin. However, this assumption does not account for *relative* changes of frequency over time. To address this issue, we consider the running mean, or the average daily query frequency across the archive, to calibrate the popularity of a query in the archive over time. For this, we reduce the query frequency of a day by subtracting the average daily query frequency computed up to that day. We use the reduced query frequencies to sort times into bins, so that bin b_0 will contain the days with the largest reduced frequency, b_1 will correspond to the second-largest reduced frequency, and so on.

Bump shapes (BUMP). In the above techniques, we considered windows of fixed size around each day for

6. We experimented with different values of k over the training data and showed that $k = 500$ is among the best performing values. We also found that considering all the matching documents did not perform well. Finally, we do not retrieve documents with $p(q|d) = 0$.

binning. But events, of course, vary in their duration, so now we do not force time intervals to have a fixed size. Instead, we identify continuous time intervals of variable length where the query frequency on each day is greater than the average query frequency per day in the entire collection. For each interval that we discover, we sum the daily query frequencies and organize the interval days into bins based on these sums. In the end, b_0 will contain all the days in intervals with the largest bumps, b_1 will correspond to intervals with the second-largest bumps, and so on.

We can use any of the above techniques for Step 2 of our algorithm; we will evaluate their relative merits in Section 5.4.

3.3.3 Computing Temporal Probability Values

In Step 3 of our algorithm, we define the $p(q|t)$ values based on the assignment of times to bins $b_0, \ldots, b_\ell$ from Step 2. As discussed, we define the binning so that bin b_i should be associated with $p(q|t)$ values that are higher than those for bin b_j whenever $i < j$. Note that a bin does not necessarily correspond to a single, uninterrupted time range. Now, we determine the final "priority level" of each time t bin using a distribution function F of our choice that depends on the bin, as follows:

$$p(q|t) = F(bin(t)),$$

where $bin(t)$ returns the index of the time t bin (see Section 3.3.2). Following Li and Croft's work on recency queries [2], we can reduce the estimated relevance of bins exponentially with their distance to the time(s) of interest, and define:

$$p(q|t) = \frac{\lambda \cdot \exp(-\lambda \cdot bin(t))}{\sum_{t_i \in dates(D)} \lambda \cdot \exp\left(-\lambda \cdot bin(t_i)\right)},$$

where λ is the parameter of the exponential distribution, often called the rate parameter, and $dates(D)$ is the time span of the news archive D, as before.

Next, we discuss yet another approach to estimate $p(t|q)$, with a focus on efficiency.

3.4 Estimation Using Word Tracking

So far, the techniques for estimating $p(t|q)$ that we have discussed in Sections 3.2 and 3.3 relied on obtaining the top-k matching documents for q as a first step, typically with $k \geq 500$. This step is unnecessarily time consuming when performed on top of an unmodified search engine, since some of the processing needed to answer a search-engine query, such as snippet generation, is not required to compute $p(t|q)$. Furthermore, query processing in a state-of-the-art search engine is often optimized to return the top-10 results and is not efficient for producing a larger set of results. In this section, we investigate an alternative and efficient technique that relies on a word temporal-tracking index that is updated as new documents are created and discovered.

The strategies in Sections 3.2 and 3.3 rely on analyzing the distribution of matching documents for a query q across time to estimate $p(t|q)$ for each t, a potentially expensive proposition. To estimate $p(t|q)$ efficiently, we refine $p(q|t)$ one step further and assume independence between query terms, to get

$$p(t|q) \propto p(t) \cdot p(q|t) \approx p(t) \cdot \prod_{w \in q} p(w|t), \quad (8)$$

where $p(w|t)$ is the probability of generating query word w at day t.

To capture the fact that news topics sometimes evolve over a period of several days, we apply a similar step as in Section 3.2 for smoothing using the simple moving-average technique for adjacent days, as follows:

$$\bar{p}(t|q) \approx \frac{1}{x} \cdot \sum_{i=0}^{x-1} p(t-i|q), \quad (9)$$

where x is the number of past days for smoothing. Of course, we can also use smoothing strategies similar to what we used in Section 3.3. For instance, we can follow the WIN strategy, which examines past and future days, as follows:

$$\bar{p}(t|q) \approx \frac{1}{2 \cdot x + 1} \cdot \sum_{i=-x}^{x} p(t+i|q), \quad (10)$$

where x is the number of past and future days for smoothing.

To efficiently compute $p(w|t)$, we now describe a word temporal-tracking index that supplies and maintains $p(w|t)$. Specifically, this index stores the values $p(w|t)$ for all terms in the collection's vocabulary and for each day t. The index can be built and maintained in different ways. The simplest way is to associate each word w with an inverted list of the days in which w appeared, including the document frequency of w in each day. These frequencies can be used to compute $p(w|t)$ according to (8). We leverage this simple approach by using Indri to build an index over a new "collection" consisting of one "large" document per day. Specifically, for each day t, we conceptually create a large document L_t that contains the concatenation of all the documents produced on day t. From the index, we then know $p(w|L_t)$ for each word w and each time t. Note that this index is small, because it covers just one (large) document per day. With this index, we can now efficiently estimate $p(w|t)$ to be $p(w|L_t)$. In other words, since $p(w|t)$ is the probability of generating the word w from the documents published at day t and is computed based on the frequency of w in these documents, we can estimate $p(w|t)$ by just using the probability $p(w|L_t)$ of generating word w from L_t, which contains all the text of these documents.

In fact, we can now directly use our special index and Indri to compute the term $p(q|t)$ from (8), rather than $p(w|t)$. We can then ignore the word independence assumption and rely on Indri, over the special index, to compute $p(q|L_t)$ and use it as an estimate of $p(q|t)$.

Now, after exploring techniques for estimating $p(t|q)$, we investigate alternatives to integrate $p(t|q)$ into several state-of-the-art retrieval models.

4 Integrating Temporal Relevance in Search

In Section 2, we discussed the general family of time-sensitive queries and claimed that "traditional" information retrieval engines do not take temporal relevance into account when answering these queries. For example, the top-10 results that The New York Times search engine returns for the time-sensitive query *[Madrid bombing]*, as of

24 March 2009, are recent articles, not including any articles from the time of the actual bombing of the Madrid trains system in 2004. In Section 3, we investigated three techniques to estimate $p(t|q)$, the temporal relevance of time t for a given query q. In this section, we naturally integrate $p(t|q)$ into several retrieval models. We use $p(t|q)$ to adjust the document relevance scores by boosting the scores of documents published within important intervals for q. Specifically, Section 4.1 discusses an approach from the literature for answering recency queries (see Section 1). Then, Sections 4.2 through 4.4 present our work for integrating $p(t|q)$ into several state-of-the-art retrieval models, to answer all time-sensitive queries.

4.1 Background: Answering "Recency" Queries

Sometimes queries issued over a news archive are after recent events or breaking news, as we discussed in Section 1. Li and Croft [2] developed a time-sensitive approach for processing recency queries. Their approach processes a recency query by computing traditional topic relevance scores for each document, and then "boosting" the scores of the most recent documents, to privilege recent articles over older ones. Language models [9] have been used as a successful approach to rank documents in a collection according to their topic relevance for a query. To estimate the relevance of a document d to a query $q, p(d|q)$, the conditional probability that d is topically relevant to q is computed. This retrieval model defines $p(d|q)$ as being proportional to $p(d) \cdot p(q|d)$, where $p(d)$ is the prior probability that d is relevant, and $p(q|d)$ is the probability that query q will be generated from document d.[7] In the original language models and in later modifications, the prior $p(d)$ is ignored since it is assumed to be uniform and constant for all documents. For recency queries, Li and Croft suggest modifying $p(q|d)$ to combine two elements, time relevance and topical relevance. Specifically, Li and Croft define the prior $p(d)$ of document d as a function of the document creation date, so that recent documents are given a greater prior value than older documents.

4.2 Answering Time-Sensitive Queries with Language Models

In contrast to recency queries, for a general time-sensitive query, we do not know beforehand either the relevant time periods or the expected distribution of relevant documents over time. Therefore, we cannot directly estimate the prior $p(d)$ of document d as described in Li and Croft [2] for recency queries. Now, we show how to integrate the temporal relevance $p(t|q)$ of Section 3 into the framework of language models [9], so that we can answer general time-sensitive queries.

Language models are a state-of-the-art general approach for ranking documents in a collection according to their topic similarity with a query. In particular, the query likelihood model [9], [10] estimates the relevance of a document d to a query q by computing the conditional probability $p(d|q)$ that d is topically relevant to q, which is defined as

$$p(d|q) \propto p(d) \cdot p(q|d). \tag{11}$$

We can read this expression as indicating that $p(d|q)$ is proportional to $p(d) \cdot p(q|d)$, where $p(d)$ is the prior probability that d is relevant and $p(q|d)$ is the likelihood that query q will be generated from document d.[8]

To answer general time-sensitive queries, we want to identify not just the relevant documents for the query, but also the relevant time periods. Craswell et al. [11] introduced a framework to complement the topical relevance of a document for a query with additional evidence (e.g., PageRank [12] or ClickDistance [11]). We build on this framework and on the idea of splitting a document d into a content component c_d as well as a temporal component t_d.[9] We can then write $p(d|q)$ as $p(c_d, t_d|q)$, which expresses the probability that c_d is topically relevant to q and that t_d is a time period relevant to q, where c_d is the content of the document d and t_d is the time when d was published. Using basic probability rules, we have

$$p(d|q) = p(c_d, t_d|q) = p(c_d|q) \cdot p(t_d|c_d, q).$$

To handle the term $p(t_d|c_d, q)$ in the equation, we can ignore c_d since the temporal relevance of day t_d for query q does not depend on the content of one particular document c_d but rather, as we have seen in Section 3, on the density of relevant documents around time t_d. As a result, we get

$$\begin{aligned} p(d|q) &\propto p(c_d|q) \cdot p(t_d|q) \\ &\propto p(q|c_d) \cdot p(c_d) \cdot p(t_d|q). \end{aligned}$$

Note that c_d is what we traditionally refer to as d in language models and our use of c_d is to emphasize that a document in our modified model consists of the traditional textual content component c_d and the temporal information t_d. The document prior $p(c_d)$ is typically assumed to be uniform for all documents, considering that there is no document that is more likely to be relevant *across all possible queries*. (However, some techniques use query-independent metrics, such as PageRank [12], to assign higher prior values $p(d)$ to some of the documents in a collection.) The term $p(q|c_d)$ corresponds to the probability of generating query q from document text component c_d and can be computed using existing techniques, such as the QL model [9], [10] or the relevance language model [13]. Finally, $p(t_d|q)$ is in fact the temporal relevance $p(t|q)$ from Section 3. As a result, we can further develop $p(c_d, t_d|q)$ as follows:

$$p(d|q) = p(c_d, t_d|q) \propto p(q|c_d) \cdot p(c_d) \cdot p(q|t_d) \cdot p(t_d),$$

where both $p(q|t_d)$ and $p(t_d)$ are as discussed in Section 3. The factor $p(q|t_d)$ corresponds to the probability of "observing" q in the documents published in time t_d, while the time prior $p(t_d)$ can be defined proportionally to the *total* number of documents published at time t_d. In this case, since the volume of news is typically lower during the weekends, $p(t_d)$ would be lower if t_d corresponds to a weekend than if it corresponds to a weekday.

Next, we show how to incorporate $p(t|q)$ as another source of relevance evidence in the probabilistic relevance model.

7. $p(q)$ is constant across documents.

8. $p(q)$ is constant across documents.

9. Note that Craswell et al. [11] choose to split the document into a content component and an additional score component, such as PageRank, whereas our second component is actual temporal information associated with the document.

4.3 Answering Time-Sensitive Queries with BM25

In the previous section, we showed how to integrate the temporal relevance $p(t|q)$ into language models. We now describe a similar integration into the probabilistic relevance model, a leading state-of-the-art approach suggested by Robertson et al. [6], [7], [14]. In defining PRM, Robertson et al. state the following principle: *"To produce the optimal ranking of a set of documents as an answer for a query at hand, the documents should be ranked by the posterior probability of belonging to the relevance class R of the query."* According to this principle, Robertson et al. showed that ranking the documents by the odds of their being observed in R produces the optimal ranking, and introduced the following general PRM framework:

$$p(R|d,q) \propto_q \log\frac{p(R|d,q)}{p(\overline{R}|d,q)} \propto_q \log\frac{p(d|R,q)}{p(d|\overline{R},q)}, \tag{12}$$

where $\overline{R}$ is the irrelevant class and $\propto_q$ implies same rank order, as usual. In the last step, Bayesian inversion is applied.

BM25 [4], [5], [6], [7], [14], [15], [16], the basic version of PRM, starts from this probability ranking principle and a set of discrete attributes that are present in the document, and defines a general ranking model as a sum of weights over discrete attributes, as follows:

$$\begin{aligned}\log\frac{p(d|R,q)}{p(D|\overline{R},q)} \propto_q BM25(d,q) &\approx \prod_V \frac{p(tf_i|R,q)}{p(tf_i|\overline{R},q)}\\ &\approx \prod_{qt} \frac{p(tf_i|R,q)}{p(tf_i|\overline{R},q)}\\ &\propto_q \sum_{qt} \log\frac{p(tf_i|R,q)}{p(tf_i|\overline{R},q)}\\ \cdots &\propto_q \sum_{qt,tf_i>0} \log W_i(tf_i),\end{aligned} \tag{13}$$

where $V = \{w_1, \ldots, w_{|V|}\}$ is the set of words that appear in database D, tf_i is the frequency of word w_i in D, and qt is the set of words in query q. In the first step, word independence is assumed. In the second, only query words are used. Finally, log is applied for the linear combination of the odds of relevance of each tf_i, which is approximated by the sum over the query word weight functions $W_i(tf_i)$. Typically, $W_i(tf_i)$ is defined by the specifics of the model that follows this principle (e.g., BM25) and is tuned over a set of queries.

The original BM25 model does not handle nontextual features of the documents. Following Craswell et al. [11] once again, we revisit (12) and introduce c_d and t_d for a document d (see Section 4.2) as follows:

$$\begin{aligned}\log\frac{p(d|R,q)}{p(d|\overline{R},q)} &= \log\frac{p(c_d,t_d|R,q)}{p(c_d,t_d|\overline{R},q)}\\ &= \log\frac{p(c_d|R,q)}{p(c_d|\overline{R},q)} + \log\frac{p(t_d|c_d,R,q)}{p(t_d|c_d,\overline{R},q)}\\ &\propto_q BM25(c_d,q) + \log\frac{p(t_d|c_d,R,q)}{p(t_d|c_d,\overline{R},q)},\end{aligned} \tag{14}$$

where $BM25(c_d,q)$ is the same as $BM25(d,q)$ in (13). By doing so, we linearly combine BM25, in a similar manner as in [11], with a term that depends on modeling time with respect to c_d, R, and q.

To handle the second term in the equation, we can ignore c_d since the temporal relevance of day t_d for query q does not depend on the content of one particular document c_d but rather, as we have seen in Section 3, on the density of relevant documents around time t_d. Based on this observation, we can use temporal relevance as follows:

$$\begin{aligned}\log\frac{p(d|R,q)}{p(d|\overline{R},q)} &\propto_q BM25(c_d,q) + \log\frac{p(t_d|R,q)}{p(t_d|\overline{R},q)}\\ &\propto_q BM25(c_d,q) + \log\frac{p(t_d|q)}{1-p(t_d|q)}.\end{aligned} \tag{15}$$

Another possibility that we explore here is to directly follow Craswell et al. [11] and separate each document d into two components, namely, a content c_d and a "score" s_d. In our scenario, the score for a document is the actual temporal relevance score of the publication date of the document. According to Craswell et al., we can then rewrite (12) as follows:

$$\begin{aligned}\log\frac{p(d|R,q)}{p(d|\overline{R},q)} &= \log\frac{p(c_d,s_d|R,q)}{p(c_d,s_d|\overline{R},q)}\\ &= \log\frac{p(c_d|R,q)}{p(c_d|\overline{R},q)} + \log\frac{p(s_d|c_d,R,q)}{p(s_d|c_d,\overline{R},q)}\\ &\propto_q BM25(c_d,q) + \log\frac{p(s_d|R,q)}{p(s_d|\overline{R},q)}\\ &\propto_q BM25(c_d,q) + w\cdot\frac{s_d^\alpha}{k^\alpha + s_d^\alpha},\end{aligned} \tag{16}$$

where we assume that the score s_d is independent of the content c_d. For the second term, Craswell et al. suggest learning a transformation function that converts the score s_d into a relevance weight. Craswell et al. evaluated several functions for different nontextual feature scores, and showed that a function of the form $w\cdot\frac{s_d^\alpha}{k^\alpha+s_d^\alpha}$, known as *sigm*, performed the best [11]. We adopt *sigm* as our transformation function of choice and tune it in Section 5.3.

4.4 Answering Time-Sensitive Queries with Pseudorelevance Feedback

We now discuss our efforts to integrate temporal relevance into a pseudorelevance feedback technique. Specifically, we focus on Indri's pseudorelevance feedback technique, which is an adaptation of Lavrenko and Croft's relevance models [13]. In the first stage of this technique, a baseline retrieval is performed to identify the top-k documents for a query at hand. According to Lavrenko and Croft, these top-k documents are then used to analyze the universe of unigram distributions and estimate $p(w|R)$, the probability that a word w appears in a document that is relevant to the query. This estimated probability is used to select the top-m representative words or phrases that are most related to the query. In the second stage, a second retrieval with query expansion is performed using the identified words or phrases.

To integrate time into this pseudorelevance feedback technique, we can account for time by biasing, in an appropriate manner, the choice of the top-k documents that are used in the first stage of query processing. We experimented with several ways to bias this choice of top-k documents based on time, but unfortunately none of these variations resulted in any gain in result accuracy for time-sensitive queries. It seems that a technique that

reranks the most topically relevant documents, and then mines them for other words related to the query, does not discover any useful words that were not already captured by the original technique. The cause of this shortcoming is most likely that the techniques are limited by their exposure to the same sets of documents. For brevity, we omit any further discussion of this technique.

5 Experiments

We now report our experimental results. Section 5.1 describes our data collections and queries. Then, Section 5.2 summarizes our experimental settings. Finally, Sections 5.3 and 5.4 discuss the results on the training and test data, respectively.

5.1 Collections and Queries

Our experiments require a large archive of news stories spread over several continuous years, as well as queries over the archive with corresponding relevance judgments. We used two document collections and two query sets, as we describe next:

5.1.1 TREC News Archive (TREC)

This collection is a portion of the TREC volumes 4 and 5, with news articles from the Financial Times (1991, 1992, 1993, and 1994) and the Los Angeles Times (1989 and 1990). We excluded the documents from the Congressional Record of the 103rd Congress (1993) and the Federal Register (1994) because these documents were not time stamped.

5.1.2 TREC Time-Sensitive Queries (TQ301, TQ351, and TQ401)

We use the subset of TREC ad hoc title queries 301-350, 351-400, and 401-450 that we identified manually as being time-sensitive queries, according to the following simple criteria. First, we discard all queries with fewer than 20 matching documents: it would not be meaningful to analyze time-distribution patterns over fewer than 20 documents. (We use conjunctive Boolean semantics for matching, following our approach in Section 3.3.1.) Second, we manually examine the title, description, and narrative of each of the remaining queries, and classify a query as time sensitive if it targets documents that are associated with specific news events (see Definition 1). For these queries, the relevant documents are not spread uniformly over time but rather tend to concentrate in restricted time intervals. When the information about a query is not sufficient to make the classification decision, we build and analyze a relevant-document histogram, using the human relevance judgments from TREC. After this manual procedure, the time-sensitive queries form three query sets, namely, **TQ301**,[10] **TQ351**,[11] and **TQ401**.[12] **TQ301** is used for training and includes 31 time-sensitive queries out of the TREC ad hoc queries 301-350. **TQ351** and **TQ401** are used for testing and include 25 and 30 time-sensitive queries from TREC ad hoc queries 351-400 and 401-450, respectively. As an alternative to this manual identification of the time-sensitive queries for our experiments, we could have used the classification strategy in [1]. (See Section 6.) We did not follow this alternative approach to be able to crisply evaluate our retrieval approach without potentially introducing query classification errors into the picture.[13]

As a baseline, we also considered running our algorithms on all queries, without any query filtering (manual or automatic). The results indicated that the performance of our system was statistically indistinguishable from the unmodified, baseline systems. Therefore, we expect the performance of a system that includes an automatic query classification component to be close to a linear combination of the results with "perfect" query classification accuracy and the current baseline results, with the exact performance values depending on the accuracy of the classification algorithm. An almost perfect classification algorithm will replicate the results that we report in this paper, while a random classifier will result in performance close to the one of the baseline. (Jones and Diaz [1] report that their temporal query classifier has accuracy around 70 to 75 percent.)

5.1.3 Newsblaster Archive (BLASTER)

This collection includes the six-year archive of Newsblaster [17], with news articles crawled daily from 25 news sources from September 2001 to December 2006.

5.1.4 Newsblaster Time-Sensitive Queries (TQBLASTER)

Unlike TREC, Newsblaster does not provide queries or relevance judgments. To build a query set for Newsblaster, we recruited five journalists who volunteered their daily queries issued on news archive engines. We gathered 125 queries and followed the same guidelines as for the TREC queries to identify the time-sensitive queries among the 125 queries. We refer to the set of 76 time-sensitive queries that we gathered as the **TQBLASTER** query set.

To obtain the relevance judgments for our queries, we launched a large-scale user study using the Amazon Mechanical Turk service. In our study, each Mechanical Turk annotator was presented with queries and associated documents, and decided on the relevance of each document for each query, following TREC-style guidelines. To ensure the quality of this distributed labeling effort, annotators had to pass a *qualification test*. Specifically, each prospective annotator was presented with a TREC query and seven TREC documents, and was only allowed to proceed with the **TQBLASTER** labeling task if he or she agreed with the TREC annotations of at least six of the seven documents in the qualification test. (The queries and documents of the qualification test were picked randomly for each prospective annotator, from among 10 TREC queries and 70 TREC documents.) Also, an annotator was allowed to label at

10. **TQ301** queries are: 301, 302, 306, 307, 311, 313, 315, 316, 318, 319, 320, 321, 322, 323, 324, 326, 329, 330, 331, 332, 333, 334, 337, 340, 341, 343, 345, 346, 347, 349, 350.

11. **TQ351** queries are: 352, 354, 357, 358, 359, 360, 366, 368, 372, 374, 375, 376, 378, 383, 385, 388, 389, 390, 391, 392, 393, 395, 398, 399, 400.

12. **TQ401** queries are: 401, 402, 404, 407, 408, 409, 410, 411, 412, 418, 420, 421, 422, 424, 425, 427, 428, 431, 432, 434, 435, 436, 437, 438, 439, 442, 443, 446, 448, 450.

13. Unfortunately, we could not use the hand-labeled queries by Jones and Diaz [1] because they considered different collections and queries for their work.

most seven **TQBLASTER** queries before being required to take another qualification test.[14]

For each query, we created a pool of the top-20 results from six retrieval techniques: four time-sensitive techniques that performed well on the TREC data along with two baseline techniques (see Section 5.4). On average, for each query, the union of the six sets of (up to) 20 documents from each technique consists of 39 unique documents. We used three annotators to get relevance judgments for each query-document pair and we accepted the annotations only when there was perfect agreement across the annotators (this was the case for 61.25 percent of the instances). In the case of disagreement, we asked for two additional annotations for each query-document pair and we took the majority opinion as the final label for such pairs. In total, 371 human subjects passed at least one of the qualification tests, and 136 of them actually participated in the study, overall producing relevance judgments for 2,822 query-document pairs.

5.2 Experimental Settings

We now elaborate on our experimental settings, which include the techniques that we compare, the associated parameters, and the metrics for our evaluation. All the techniques listed below have two versions, based on the query likelihood [9] and the relevance models [13]. We implemented all techniques using Indri and Lemur. All documents and queries were stemmed using the Krovetz stemmer [18] and stopped using Indri's and Lemur's stop words list. To experiment with the time-sensitive BM25-based techniques of Section 4.3, we use Lemur's implementation of BM25.

5.2.1 LM-Based Baseline Techniques

We experiment with state-of-the-art versions of topic-based retrieval techniques, namely, QL and RM (see Section 4.2), to which we refer as QL-TOPIC and RM-TOPIC, respectively. We also evaluate our techniques against two versions of Li and Croft's strategy for recency queries [2] (see Section 6), one for QL and one for RM, to which we refer as QL-RECENCY and RM-RECENCY, respectively.

5.2.2 BM25-Based Baseline Techniques

We experiment with Lemur's implementation of BM25 and refer to it as BM25-TOPIC.

5.2.3 Time-Sensitive Language Model Variations (Section 4.2)

Our time-sensitive approach for language models has several parameters, including the histogram construction strategies, with DAY, FIXED, WIN, MEAN, and BUMP as options (Section 3.3.2), the parameters of each histogram construction strategy, such as parameter x of WIN and FIXED, and parameter λ of the distribution function for the time periods (Section 3.3.3). We experimented with the five possible histogram construction strategies—with different settings for each—and with alternate parameter values. This resulted in 18 variations of our time-sensitive approaches: nine variations have QL as the underlying retrieval model and various values for x, shown in Table 1; the other nine variations have RM as the underlying retrieval model. In Section 5.3, we will discuss how we set the value of λ over the training set. For our experiments, and for simplicity, we assume that the time prior $p(t)$ is constant across the different values of t (see Section 4.2).

14. This requirement ensures that annotators that contribute a large number of judgments also pass more qualification tests.

TABLE 1
The QL-Based Techniques that We Compared Experimentally (We Also Evaluated the RM-Based Counterparts)

Name	*Description*
SUM-QL	Direct estimation of $p(t\|q)$ with $x = 14$
DAY-QL	DAY binning estimation of $p(t\|q)$
TEN-QL	FIXED binning estimation of $p(t\|q)$ with $x = 10$ days
MONTH-QL	FIXED binning estimation of $p(t\|q)$ with $x = 30$ days
WIN3-QL	WIN binning estimation of $p(t\|q)$ with $x = 3$ days
WIN7-QL	WIN binning estimation of $p(t\|q)$ with $x = 7$ days
WIN14-QL	WIN binning estimation of $p(t\|q)$ with $x = 14$ days
WIN28-QL	WIN binning estimation of $p(t\|q)$ with $x = 28$ days
MEAN-QL	MEAN binning estimation of $p(t\|q)$
BUMP-QL	BUMP binning estimation of $p(t\|q)$
WORD-QL	Word tracking estimation of $p(t\|q)$ with $x = 14$
QL-RECENCY	Recency baseline [2]
QL-TOPIC	Topic-similarity baseline [9]

To explore the impact of our binning technique of Section 3.3, we consider an alternative definition $p(t|q)$ that does not rely on binning. Specifically, we leverage recent work by Jones and Diaz on analyzing temporal characteristics of queries [1] (see Section 6) and define

$$p(t|q) = \Sigma_{d \in R} p(t|d) \cdot \frac{p(d|q)}{\Sigma_{\hat{d} \in R} p(\hat{d}|q)},$$

where $p(t|d) = 1$ if d was published on day t and 0 otherwise, and R is a set with the top-k ranked documents for q, according to the QL model, which is used to compute the $p(d|q)$ values (see Section 5.3). For our experiments, and following [19], we set $k = 1{,}000$. The $p(t|q)$ values are then normalized using a background probability of the collection and smoothed using an average of 14 days [1]. We refer to the alternative technique that results from this definition of $p(t|q)$ as SUM-QL. We also consider the corresponding version using RM instead of QL, and refer to it as SUM-RM.

5.2.4 Time-Sensitive BM25 Variations (Section 4.3)

In the first time-sensitive BM25 variation, which we refer to as DIRECT-BM25, we compute the time-based relevance weight as in (15) using the temporal relevance values as described in Section 4.3. In the second BM25 variation, which we refer to as APPROX-BM25, we estimate the time-based relevance weight as in (16), using a *sigm* transformation function [11] with parameters w, k, and α, which we will determine in Section 5.3.

5.2.5 Index-Based Time-Sensitive Computations (Section 3.4)

We explore the impact of computing $p(t|q)$ efficiently using our temporal word tracking index of Section 3.4. We refer to the search techniques that use this method as WORD-QL and WORD-RM.

TABLE 2
Top Performing QL- and RM-Based Techniques versus Baseline Alternatives over Training Data

Technique	QL				RM			
	λ	MAP	$P@10$	$P@20$	λ	MAP	$P@10$	$P@20$
SUM	n/a	0.140	0.396	0.364	n/a	0.182	0.461	0.389
DAY	0.005	0.203	0.459	0.416	0.001	0.217	0.441	0.408
TEN	0.02	0.202	0.463	0.417	0.001	0.217	0.441	0.408
MONTH	0.005	0.203	0.463	0.416	0.020	0.218	0.447	0.409
WIN7	0.05	0.2023	0.456	0.419	0.050	0.217	0.441	0.409
MEAN	0.005	0.203	0.459	0.419	0.0005	0.217	0.441	0.408
BUMP	0.01	0.202	0.459	0.417	0.001	0.217	0.441	0.408
WORD	n/a	0.204	0.470	0.421	n/a	0.208	0.471	0.401
RECENCY	0.02	0.161	0.447	0.394	0.02	0.164	0.431	0.397
TOPIC	n/a	0.189	0.440	0.408	n/a	0.205	0.431	0.4

5.2.6 Evaluation Metrics

We follow TREC's evaluation procedure for evaluating ad hoc queries using the trec_eval package.[15] This package reports *precision* and *recall* at different levels for each query. It also outputs summary measures across all queries. We report results for the following measures [20]:

1. The "recall level precision averages" table, which contains the average precision at 11 standard recall levels (0 to 1 in increments of 0.1). We only consider precision at the top recall cutoffs, namely, 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5 ($P@0, P@0.1, P@0.2, P@0.3$, $P@0.4$, and $P@0.5$, respectively).
2. The recall-precision graph, which plots the previous table.
3. The mean (noninterpolated) average precision (MAP), which is the average of the precision value obtained after each relevant document is retrieved.
4. The "document level precision" table, which contains the average precision at nine document levels; we consider precision at 5, 10, 15, and 20 documents ($P@5, P@10, P@15$, and $P@20$, respectively).
5. The average R-Precision (RAP), which is the precision after R documents has been retrieved and R is the number of relevant documents.

5.3 Results on Training Queries

In this section, we describe our choice of parameters for our techniques over the training queries, namely, over the **TQ301** query set.

5.3.1 Tuning the Language Model Variations

To determine the window size x for WIN (Section 3.3.2), we experimented with values 3, 7, 14, and 28. Surprisingly, the window size did not make any significant difference for our performance metrics across several different λ values. This is mainly due to the bursty posting of news documents: typically, there is high activity around the date of an event, and minimal posting activity in other days. In our test set, we did not have cases where two distinct events, corresponding to the same query, overlapped within a time period of 28 days. So for the rest of our experiments, we set the window size $x = 7$ (i.e., we use the WIN7-QL and WIN7-RM versions for WIN). In the cases where distinct events may overlap within periods of seven days (e.g., news about the games of a sports team), it would be advisable to keep the window time small, to better capture the distinct nature of the events.

15. http://trec.nist.gov/trec_eval/.

TABLE 3
The Performance of DIRECT-BM25 for Different Temporal Relevance Variants

Technique	MAP	$P@5$	$P@10$	$P@20$
SUM	0.1108	0.4021	0.3823	0.3256
MEAN ($\lambda = 0.005$)	0.1136	0.4187	0.3969	0.3359
BUMP ($\lambda = 0.01$)	0.1116	0.4191	0.3942	0.3321
WORD	0.1142	0.4039	0.3806	0.3243

We now turn to choosing the λ value for each binning technique (Section 3.3.3). For this, we ran our techniques over the **TQ301** training queries with 12 different values of λ, from 0.0001 to 0.2, and using QL. For brevity, we do not show the performance of our techniques across these values. Also, we only report a portion of the trec_eval output. In general, for all techniques except WIN7-QL,[16] decreasing λ up to a certain value improves the result quality, but quality either drops or stays constant after that. Since high values of λ eliminate differences between time periods, this illustrates that focusing on specific time periods is important. However, very small values of λ decrease rapidly the relevance scores for documents that are not in the top bin, and equates the scores of these (relevant) documents with the scores of documents that are nonrelevant.

For each technique, we checked the statistical significance (using Wilcoxon's signed rank test [21]) for MAP with λ equal to 0.2, 0.01, and 0.005, and found $\lambda = 0.01$ or $\lambda = 0.005$ are generally better than $\lambda = 0.2$ and that the difference is statistically significant ($p < 0.001$). We also verified similar observations for $P@20$ and MAP. We ultimately chose to select the parameter values based on $P@20$, then $P@10$, and finally using MAP to break ties. Table 2 summarizes the results for our techniques with the best λ values as well as the results for the baselines. All the QL-based binning techniques improve MAP by 7 percent against QL-TOPIC and by 25 percent against QL-RECENCY over the training queries, a statistical significant difference ($p < 0.001$, Wilcoxon's signed rank test). The results are similar for the RM-based binning techniques. For our test data, we choose to run our experiments for BUMP-QL, MEAN-QL, and TEN-QL, for the QL model, and for MONTH-RM, BUMP-RM, and TEN-RM, for the RM model, with the λ values in Table 2.

5.3.2 Tuning SUM-QL and SUM-RM

Analogous to our tuning of the binning techniques, we examined the performance of the SUM techniques with $p(t|q)$ computed on **TQ301** using the QL model with and without conjunctive Boolean filtering. The results were better without the conjunctive Boolean filtering, so for our test evaluation, we do not use conjunctive Boolean semantics.

5.3.3 Tuning the BM25 Variations

To keep our experiments manageable, for the BM25 variations, we first settle on the choice of temporal relevance formulation by examining the performance of the DIRECT-BM25 technique over our training set. (We focus on DIRECT-BM25 simply to reduce the amount of tuning.) Then, we decide on the parameters for the *sigm* function that we use in APPROX-BM25. Table 3 shows the performance of DIRECT-BM25 using four different temporal relevance

16. For WIN7-QL, the λ value seems to make no difference in performance for the values that we examined.

TABLE 4
Top-10 Performing Settings for APPROX-BM25, for $P@20$ and over Training Data

k	w	α	*MAP*	$P@5$	$P@10$	$P@20$
0.00002	8	1.6	0.1128	0.4312	0.3906	0.3516
0.00002	8	1.5	0.1129	0.4312	0.3906	0.3484
0.00002	8	1.7	0.1126	0.4312	0.3906	0.3484
0.000015	8	2	0.1118	0.425	0.4	0.3469
0.00002	8	1.4	0.1129	0.4312	0.3906	0.3469
0.00005	16	1	0.1126	0.4312	0.3906	0.3469
0.000015	8	1.4	0.1123	0.425	0.3875	0.3453
0.000015	8	1.7	0.112	0.425	0.3906	0.3453
0.000015	16	0.5	0.1121	0.425	0.3844	0.3453
0.00002	8	2	0.1124	0.4312	0.3969	0.3453

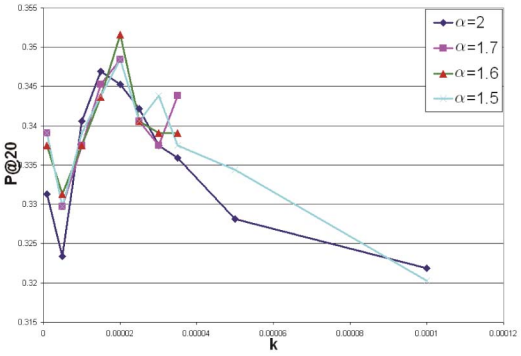

Fig. 4. $P@20$ values for APPROX-BM25, over the training data set, for $w = 8$ and across several k and α values.

TABLE 5
Performance of QL-Based Techniques over the **TREC45** Data Set for the **TQ351** Query Set

Metric	*Baselines*		*Time-Sensitive Algorithms*				
	TOPIC	*RECENCY*	*SUM*	*WORD*	*BUMP* $\lambda = .01$	*MEAN* $\lambda = .005$	*TEN* $\lambda = .02$
Rel	3350	3350	3350	3350	3350	3350	3350
Rret	1371	1090	1379	1463	1481	1478	1482
P@0.0	0.596	0.613	0.678(+)	0.714(+)	**0.747**(+)	**0.739**(+)	**0.747**(+)
P@0.1	0.327	0.344	0.348	**0.392**	0.354	0.352	0.354
P@0.2	0.226	0.204	0.213	0.231	**0.267**(+)	0.266(+)	0.264(+)
P@0.3	0.185	0.140	0.151	0.191	**0.208**(+)	0.207(+)	**0.208**(+)
P@0.4	0.142	0.096	0.092	0.121	**0.158**	**0.158**	0.157
P@0.5	0.090	0.041	0.060	0.093	0.114	**0.115**	0.114
MAP	0.127	0.117	0.121	**0.150**(+)	**0.148**(+)	**0.149**(+)	**0.148**(+)
P@5	0.384	**0.520**(+)	0.429(+)	**0.518**(+)	0.464(+)	0.464(+)	0.464(+)
P@10	0.368	0.428(+)	0.388	**0.462**(+)	**0.452**(+)	**0.452**(+)	0.448(+)
P@15	0.363	0.429(+)	0.349	**0.429**(+)	0.400(+)	0.403(+)	0.397(+)
P@20	0.340	**0.382**	0.318	0.367	0.376	0.374	0.372
RAP	0.188	0.181	0.169	0.205(+)	0.209(+)	**0.210**(+)	0.209(+)

The + sign indicates performance that is better, in a statistically significant manner, than the QL-TOPIC baseline. (There were no cases of a time-sensitive algorithm having lower performance in a statistically significant manner.) The boldfaced numbers mark the techniques with the highest performance for the particular metric. (Multiple boldfaced numbers indicate that the performance of the algorithms is statistically indistinguishable.)

formulations, namely, SUM, MEAN, BUMP, and WORD. While all DIRECT-BM25 variations have a different performance in terms of $P@10$ and $P@20$, they all have similar MAP values. Since MEAN and BUMP maintain a slight advantage over WORD and SUM, we use MEAN for the rest of our evaluation.

We believe that this advantage is attributable to the fact that both the MEAN and BUMP values are determined by a distribution function, unlike the WORD and SUM values. For example, the probability values according to MEAN and BUMP are distributed exponentially, whereas that is not the case for the WORD and SUM values. This difference is critical for DIRECT-BM25 since we use these temporal relevance values to directly estimate, by computing the odds (15), the relevance weights of the time component in the modified BM25. This is not as critical for APPROX-BM25, since it uses a *sigm* transformation function to convert the temporal relevance values to relevance weights.

Finally, APPROX-BM25 uses a transformation function with the parameters w, k, and α (16). Via an exhaustive exploration of parameter-value combinations, we obtained $w = 8, k = 0.00002$, and $\alpha = 1.6$ as the best values. Table 4 shows the results of the top-10 performing settings, while Fig. 4 shows the tuning of k across several values of α and for $w = 8$.

TABLE 6
Performance of QL-Based Techniques over the **TREC45** Data Set for the **TQ401** Query Set, Following the Reporting Conventions of Table 5 but with Respect to the Best Performing Baseline

Metric	*Baselines*		*Time-Sensitive Algorithms*				
	TOPIC	*RECENCY*	*SUM*	*WORD*	*BUMP* $\lambda = .01$	*MEAN* $\lambda = .005$	*TEN* $\lambda = .02$
Rel	3793	3793	3793	3793	3793	3793	3793
Rret	2115	1467	2145	2301	2279	2279	2279
P@0.0	0.667	0.687	**0.728**(+)	**0.719**(+)	0.697(+)	0.696(+)	0.689(+)
P@0.1	0.408	0.412	0.374	0.419(+)	**0.429**(+)	**0.427**(+)	**0.427**(+)
P@0.2	0.339	0.285	0.246	0.341	**0.356**(+)	**0.357**(+)	**0.358**(+)
P@0.3	0.268	0.209	0.168	0.213	0.287(+)	0.288(+)	**0.291**(+)
P@0.4	0.188	0.124	0.096	0.178	**0.200**	**0.200**	**0.200**
P@0.5	0.139	0.100	0.076	0.128	0.168	0.168	**0.169**
MAP	0.182	0.154	0.133	0.194(+)	**0.202**(+)	**0.202**(+)	**0.202**(+)
P@5	0.420	0.513(+)	0.467(+)	**0.543**(+)	0.460(+)	0.460(+)	0.447(+)
P@10	0.377	**0.440**(+)	0.390	**0.463**(+)	0.407(+)	0.407(+)	0.410(+)
P@15	0.378	0.413(+)	0.356	**0.433**(+)	0.387	0.391	0.389
P@20	0.367	0.382	0.331	0.371	**0.393**	0.392	0.388
RAP	0.249	0.224	0.198	0.259(+)	**0.263**(+)	**0.263**(+)	0.262(+)

TABLE 7
Performance of RM-Based Techniques over the **TREC45** Data Set for the **TQ351** Query Sets, Following the Reporting Conventions of Table 5 but with Respect to the RM-TOPIC Baseline

Metric	*Baselines*		*Time-Sensitive Algorithms*				
	TOPIC	*RECENCY*	*SUM*	*WORD*	*MONTH* $\lambda = .02$	*BUMP* $\lambda = .0005$	*TEN* $\lambda = .001$
Rel	3962	3962	3962	3962	3962	3962	3962
Rret	2179	2179	2183	2261	2194	2244	2244
P@0.0	0.646	0.641	0.708(+)	**0.728**(+)	0.656(+)	0.671(+)	0.671(+)
P@0.1	0.446	0.455	0.457	**0.463**	0.449	**0.460**	**0.460**
P@0.2	0.362	0.338	0.374	0.374	0.365	**0.373**(+)	**0.373**(+)
P@0.3	0.309	0.235	0.292	0.319	0.312	**0.322**	**0.322**
P@0.4	0.244	0.174	0.236	0.231	0.244	**0.257**(+)	**0.257**(+)
P@0.5	0.196	0.131	0.181	0.171	0.199	**0.209**	**0.209**
MAP	0.205	0.173	0.210	0.213(+)	0.207	**0.215**(+)	0.215(+)
P@5	0.469	**0.520**(+)	**0.517**(+)	**0.521**(+)	0.450	0.475	0.475
P@10	0.444	**0.492**(+)	0.467	**0.497**(+)	0.441	0.453	0.453
P@15	0.421	0.451	0.430	**0.457**	0.415	0.419	0.419
P@20	0.406	**0.416**	0.415	0.401	0.400	**0.416**	**0.416**
RAP	0.255	0.233	0.262(+)	**0.277**(+)	0.258	0.263(+)	0.263(+)

TABLE 8
Performance of RM-Based Techniques over the **TREC45** Data Set for the **TQ401** Query Set, Following the Reporting Conventions of Table 5 but with Respect to the RM-TOPIC Baseline

Metric	*Baselines*		*Time-Sensitive Algorithms*				
	TOPIC	*RECENCY*	*SUM*	*WORD*	*MONTH* $\lambda = .02$	*BUMP* $\lambda = .0005$	*TEN* $\lambda = .001$
Rel	3793	3793	3793	3793	3793	3793	3793
Rret	2334	1555	2503	2532	2500	2507	2507
P@0.0	0.603	0.611	**0.727**(+)	0.721(+)	0.663(+)	0.648(+)	0.641
P@0.1	0.437	0.465(+)	**0.490**(+)	**0.497**(+)	0.469(+)	0.469(+)	0.468(+)
P@0.2	0.377	0.328	0.348	**0.407**(+)	0.395(+)	**0.401**(+)	**0.401**(+)
P@0.3	0.323	0.256	0.261	0.327	0.336	**0.338**	**0.338**
P@0.4	0.244	0.162	0.193	0.251	0.260(+)	**0.265**(+)	0.263(+)
P@0.5	0.186	0.111	0.151	0.195	0.217	**0.221**	0.220
MAP	0.213	0.171	0.201	**0.239**(+)	0.234(+)	**0.235**(+)	**0.235**(+)
P@5	0.380	0.473(+)	0.551(+)	**0.563**(+)	0.433(+)	0.433(+)	0.413
P@10	0.393	**0.453**(+)	0.442(+)	0.451(+)	0.423	0.420	0.423
P@15	0.389	0.416	**0.424**	**0.431**	0.400	0.407	0.407
P@20	0.380	0.408	0.395(+)	**0.400**(+)	0.388	**0.398**(+)	**0.398**(+)
RAP	0.274	0.242	0.249	**0.291**(+)	0.285(+)	**0.287**(+)	**0.287**(+)

5.4 Results on Test Queries

5.4.1 Results Using Time-Sensitive TREC Queries

Tables 5, 6, 7, and 8 summarize the recall and precision of the various QL- and RM-based techniques. In terms of MAP, BUMP-QL resulted in an improvement of 17 and 26 percent against QL-TOPIC and QL-RECENCY over

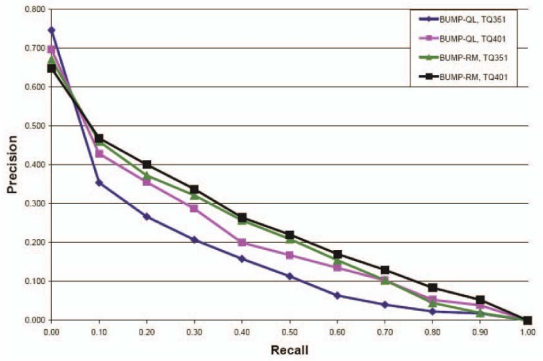

Fig. 5. Precision-recall curve for the BUMP techniques over the **TREC45** data set, for the **TQ351** and **TQ401** query sets.

TQ351, and in an improvement of 11 and 31 percent over **TQ401**. As anticipated by the training results, MEAN-QL and TEN-QL also achieve similar results. The results for the RM-based binning techniques are analogous to those for their QL-based counterparts: BUMP-RM shows a statistically significant improvement for MAP of 5 and 24 percent against the RM-TOPIC and RM-RECENCY baselines over **TQ351**, and a statistically significant improvement of 10 and 37 percent over **TQ401**. Fig. 5 shows precision-recall curves for the BUMP techniques. The results for SUM-QL and SUM-RM support our earlier observation that these techniques exhibit high precision for the top recall cutoff levels. In terms of $P@5$, SUM-QL results in an improvement of 11 percent against QL-TOPIC over **TQ351**, and in an improvement of 11 percent over **TQ401**. SUM-RM results in more consistent improvements over **TQ401** across $P@5$, $P@10$, and $P@20$.

Tables 9 and 10 summarize the recall and precision of the various BM25-based techniques. In terms of $P@20$, both DIRECT-BM25 and APPROX-BM25 resulted in an improvement of 7 percent against BM25-TOPIC over **TQ351**. DIRECT-BM25 also resulted in an improvement of 5, 6, and 3 percent against BM25-TOPIC over **TQ351** in terms of $P@5, P@10$, and $P@15$, respectively. APPROX-BM25 does not improve $P@5$ over **TQ351** against BM25-TOPIC but it improves $P@10$ and $P@15$. Similar observations follow for **TQ401**. In terms of $P@20$, DIRECT-BM25 resulted in an improvement of 8 percent against BM25-TOPIC, while APPROX-BM25 resulted in an improvement of 4 percent.

TABLE 9
Performance of the BM25-Based Techniques over the **TREC45** Data Set and for the **TQ351** Query Set, Following the Reporting Conventions of Table 5 but with Respect to the BM25-TOPIC Baseline

Metric	*TOPIC*	*DIRECT*	*APPROX*
$P@0.00$	0.6532	**0.6753**	0.6546
$P@0.10$	0.3157	0.3375	**0.3434**
$P@0.20$	0.2092	**0.2098**	0.2011
$P@0.30$	0.1114	**0.1208**	**0.1208**
$P@0.40$	0.0655	**0.0706**	0.069
$P@0.50$	**0.0292**	0.029	0.0241
MAP	0.1049	**0.1067**	0.1041
$P@5$	0.4333	**0.4571**	0.4143
$P@10$	0.3619	**0.3833**	0.3738
$P@15$	0.327	**0.3381**	0.3413
$P@20$	0.2811	**0.3012**	**0.3012**
$P@30$	0.2548	**0.2635**	0.2587
RAP	0.1593	**0.1654**	0.1641

For APPROX, we use $w = 8, k = 0.00002$, *and* $\alpha = 1.6$.

TABLE 10
Performance of the BM25-Based Techniques over the **TREC45** Data Set and for the **TQ401** Query Set, Following the Reporting Conventions of Table 5 but with Respect to the BM25-TOPIC Baseline

Metric	*TOPIC*	*DIRECT*	*APPROX*
$P@0.00$	0.659	**0.6799**	0.6712
$P@0.10$	0.3756	**0.382**	0.3698
$P@0.20$	**0.2489**	0.2388	0.2231
$P@0.30$	**0.1328**	0.1322	0.1253
$P@0.40$	0.0871	**0.0897**	0.0859
$P@0.50$	0.0668	**0.0704**	0.0655
MAP	**0.1241**	0.1235	0.1187
$P@5$	0.4614	0.4762	**0.4857**
$P@10$	0.394	**0.4**	0.3929
$P@15$	0.3251	**0.3698**	0.3587
$P@20$	0.3193	**0.3452**	0.3333
$P@30$	0.2976	**0.3016**	0.2937
RAP	0.1861	**0.1885**	0.1807

For APPROX, we use $w = 8, k = 0.00002$, *and* $\alpha = 1.6$.

Both DIRECT-BM25 and APPROX-BM25 improve $P@5$ and $P@15$ but achieve negligible or no improvement for $P@10$.

An example of a query that benefited from our techniques is *[Industrial Espionage]*. The vast majority of the relevant documents for this query are concentrated within a small time period. Our binning techniques and the SUM-based techniques capture this fact and return more documents from this period than the baseline techniques do. For example, BUMP-RM returned 19 more documents from this period than RM-TOPIC did, considering the top 100 results returned by each technique. Moreover, over the **TQ301** query set BUMP-RM returned, overall, 2,004 documents that had not been retrieved by any TREC participants for these queries and, therefore, did not have associated relevance judgments. To circumvent this problem of missing relevance judgments, we now report experimental results on the **TQBLASTER** queries, with relevance judgments gathered using Amazon's Mechanical Turk service. An advantage of this query set over the TREC set is that we obtained complete relevance judgments for the top-20 results retrieved by each of the techniques that we compare.

5.4.2 Results Using Time-Sensitive Newsblaster Queries

We now report results for **TQBLASTER**, for BUMP-QL, BUMP-RM, SUM-QL, SUM-RM, QL-TOPIC, and RM-TOPIC with the same λ values used for **TQ351** and **TQ401**. We selected the best two baseline techniques and four time-sensitive techniques according to the TREC experiments, and excluded the other techniques to keep the amount of human annotations that we needed at manageable levels. Table 11 summarizes the recall and precision values, while Fig. 6 shows precision-recall curves for the various techniques and the baselines. As an interesting observation, both BUMP-QL and BUMP-RM have improved precision at the top recall cutoff levels significantly relative to the baseline techniques QL-TOPIC and RM-TOPIC. This is also true for the SUM-based techniques. However, the precision of our techniques and the SUM-based techniques drops for higher recall cutoff levels. For example, for P@10, using BUMP-QL and SUM-QL resulted in an improvement of 31 and 32 percent, respectively, against QL-TOPIC, while BUMP-RM and SUM-RM resulted in an improvement of 29 and 27 percent, respectively, against RM-TOPIC; in contrast, P@30 values

TABLE 11
Performance over the **BLASTER** Data Set, for the **TQBLASTER** Query Set, Following the Reporting Conventions of Table 5 but with Respect to the Best Performing Baseline

	Time-Sensitive Algorithms				*Baselines*	
Metric	*BUMP-QL*	*BUMP-RM*	*SUM-QL*	*SUM-RM*	*QL-TOPIC*	*RM-TOPIC*
Rel	778	778	778	778	778	778
Rret	653	669	644	659	588	610
P@0.0	0.7116(+)	**0.7889**(+)	0.6914(+)	0.7716(+)	0.6288	0.6122
P@0.1	0.6977(+)	**0.7154**(+)	0.6834(+)	**0.7081**(+)	0.6089	0.6053
P@0.2	0.6685(+)	0.6442(+)	**0.6731**(+)	0.6211(+)	0.5869	0.5902
P@0.3	**0.6275**(+)	0.6016(+)	**0.6198**(+)	0.5901	0.5651	0.5728
P@0.4	0.5203	0.544	0.5317	0.5241	**0.5527**	0.5608
P@0.5	0.4981	0.4801	0.5011	0.4910	0.5401	**0.5525**
MAP	0.4905(+)	**0.5035**(+)	0.4878(+)	**0.4974**(+)	0.4401	0.4435
P@5	0.6082(+)	**0.6161**(+)	0.601(+)	**0.6104** (+)	0.4712	0.4849
P@10	0.5891(+)	**0.6033**(+)	0.5964(+)	0.5921(+)	0.4493	0.4658
P@15	**0.5476**(+)	0.4717(+)	**0.5481**(+)	0.4634(+)	0.421	0.4384
P@20	0.4	0.4151	0.403	0.4117	0.4027	**0.4178**
P@30	0.2667	0.2767	0.268	**0.2822**(+)	0.2685	0.2785
RAP	**0.429**	0.4211	0.4273	0.4193	0.4286	0.4142

are almost the same for all techniques. We also noticed that, for certain queries, the time-sensitive techniques introduce relevant documents that the baseline techniques could not capture: across all queries in the **TQBLASTER** query set, we counted 118 *relevant* documents that were *not* captured by QL-TOPIC or RM-TOPIC.

5.4.3 Summary

We have showed that considering time as an additional factor for ranking query results may be valuable for answering time-sensitive queries. Our results indicate that using temporal evidence derived from news archives often increases precision and reveals new relevant documents from important time intervals.

6 Related Work

Our approach expands Li and Croft's [2] strategy to process recency queries, which utilizes a language modeling framework [9], [10], [13], [22]. Most language modeling approaches assume that the prior probability $p(d)$ that a document is relevant to a query is constant. Li and Croft modified the prior $p(d)$ to reflect the fact that recently published documents are more likely to be relevant to recency queries. In our approach, which handles a broader class of time-sensitive queries, including nonrecency queries, it is not appropriate to modify the document prior $p(d)$, as we would have to introduce query-specific information (i.e., the temporal characteristics of the query) in the document prior probability $p(d)$, which is assumed to be query independent. In Section 5, we experimentally compared our techniques against Li and Croft's strategies, which we dubbed QL-RECENCY and RM-RECENCY.

Recently, Mishne [23] introduced a temporally aware technique for event search over blogs. Specifically, Mishne estimates a temporal prior for the blog postings using a histogram of the top-500 posts that are most topically relevant to a given query, similarly to the DAY technique of Section 3.3, and linearly combines this temporal prior with topical relevance to rerank the top query results. Changing the document prior probabilities has been exploited in several other contexts, such as imposing prior beliefs on the retrieval task using PageRank or inlink [11], handling webpages from different categories with appropriate prior values [24], dealing with the absence of topic [25], and so on.

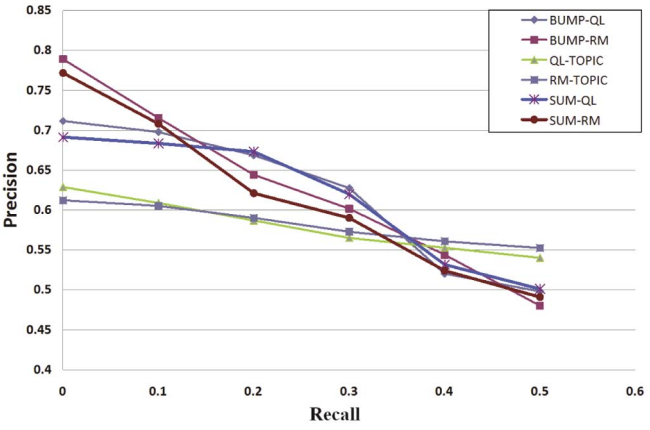

Fig. 6. Precision-recall curves over the **BLASTER** data set, for the **TQBLASTER** query set.

Time has also been used to model the evolution of topics over time. For example, Blei and Lafferty [26] introduced techniques that examine how scientific topics evolve over time. The topics, identified using a variation of Latent Dirichlet Allocation (LDA), are modeled as distributions of words. One possible extension of our technique would be to match the user query with the LDA topics instead of using a word-based language model. In this scenario, we could use our techniques together with the temporal topic frequencies identified by Blei and Lafferty's technique to locate time periods of interest. The problem of locating emerging topics [27] has also been studied extensively mainly in the context of novelty detection [28], [29], [30]. However, there is very little work on incorporating time information to improve the quality of search results.

Finally, Jones and Diaz [1], [31] analyze various temporal characteristics of queries. Specifically, they study the distribution of query results over a timeline to identify events for the queries and attempt to predict the precision of the query results. Jones and Diaz also automatically categorize queries as *atemporal*, *temporally ambiguous*, or *temporally unambiguous*. For our work, we focused on answering queries that were *manually* identified as being time sensitive (Section 2). As we discussed, we opted to use the manual classification of queries as time sensitive or not to disentangle the effects of query classification from the performance of the retrieval algorithms. In the future, we will explore combining our work with Jones and Diaz's query categorization strategy. Finally, in this paper, we experimentally explored a definition of $p(t|q)$ (see Section 5.2) that is based on Jones and Diaz's work [1]. We referred to the resulting document ranking strategies as SUM-QL and SUM-RM.

7 Conclusions and Future Work

We presented a method for processing time-sensitive queries over a news archive, with techniques for identifying important time periods for a query. We presented an extensive experimental evaluation, including TREC as well as an archive of news articles, and showed that our

techniques improve the quality of search results, compared to the existing state-of-the-art algorithms.

Our work demonstrates that integrating time in the retrieval task can improve the quality of the retrieval results, and motivates further research in the area. Currently, we rely on the publication time of the documents to locate time periods of interest. However, a document published at a later date (e.g., a review article, summarizing an event) may also be relevant; an interesting direction for future research is to infer the temporal relevance of a document by analyzing its contents [32], and not by relying solely on its publication date. Another promising research direction is to introduce time-based diversity in query results by grouping the results into clusters of relevant time ranges, enabling users to be aware of and interact with time information when examining the query results. Along the same lines, as future work, we are interested in integrating our retrieval techniques with algorithms for query reformulation, so that searchers are shown reformulations of their queries that target specific time periods, as suggested by Jones and Diaz for *temporally ambiguous* queries [1].

Another interesting direction for future work is to examine techniques that consider a time-sensitive definition of relevance at the document level. For example, queries such as [*presidential election 2012*] return documents that have varying relevance over time. For example, (relevant) documents returned today for the query [*presidential election 2012*] are unlikely to be considered highly relevant in 2012, where new information about the event will appear. Of course, handling such a time-varying definition of relevance may require extensive rethinking of the existing ways of evaluating retrieval performance, as static definitions of relevance (the case with current experimental testbeds) are not suitable for this task.[17]

Overall, we believe that seamlessly integrating temporal information into web search—for news articles or otherwise—is a promising direction for future research that can significantly improve the web search experience.

ACKNOWLEDGMENTS

Panagiotis G. Ipeirotis was supported by the US National Science Foundation (NSF) under Grant No. IIS-0643846 and by an NYU/Poly Seed Grant.

REFERENCES

[1] R. Jones and F. Diaz, "Temporal Profiles of Queries," *ACM Trans. Information Systems,* vol. 25, no. 3, article 14, 2007.
[2] X. Li and W.B. Croft, "Time-Based Language Models," *Proc. 12th ACM Conf. Information and Knowledge Management (CIKM '03),* 2003.
[3] D. Metzler and W.B. Croft, "Combining the Language Model and Inference Network Approaches to Retrieval," *Information Processing and Management,* vol. 40, no. 5, pp. 735-750, Sept. 2004.
[4] S.E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau, "Okapi at TREC," *Proc. Fourth Text REtrieval Conf. (TREC-4),* 1994.
[5] S.E. Robertson, "Overview of the Okapi Projects," *J. Documentation,* vol. 53, no. 1, pp. 3-7, 1997.
[6] K.S. Jones, S. Walker, and S.E. Robertson, "A Probabilistic Model of Information Retrieval: Development and Comparative Experiments - Part 1," *Information Processing and Management,* vol. 36, no. 6, pp. 779-808, 2000.
[7] K.S. Jones, S. Walker, and S.E. Robertson, "A Probabilistic Model of Information Retrieval: Development and Comparative Experiments - Part 2," *Information Processing and Management,* vol. 36, no. 6, pp. 809-840, 2000.
[8] W. Dakka, L. Gravano, and P.G. Ipeirotis, "Answering General Time-Sensitive Queries," *Proc. 17th ACM Conf. Information and Knowledge Management (CIKM '08),* pp. 1437-1438, 2008.
[9] J.M. Ponte and W.B. Croft, "A Language Modeling Approach to Information Retrieval," *Proc. 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '98),* 1998.
[10] F. Song and W.B. Croft, "A General Language Model for Information Retrieval," *Proc. Eighth ACM Conf. Information and Knowledge Management (CIKM '99),* 1999.
[11] N. Craswell, S.E. Robertson, H. Zaragoza, and M. Taylor, "Relevance Weighting for Query Independent Evidence," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '05),* 2005.
[12] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. Seventh Int'l World Wide Web Conf. (WWW '98),* 1998.
[13] V. Lavrenko and W.B. Croft, "Relevance-Based Language Models," *Proc. 24th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '01),* 2001.
[14] S.E. Robertson, "The Probability Ranking Principle in IR," *Readings in Information Retrieval,* pp. 281-286, Morgan Kaufmann, 1997.
[15] S.E. Robertson, S. Walker, and M. Hancock-Beaulieu, "Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive Track," *Proc. Seventh Text REtrieval Conf. (TREC-7),* 1998.
[16] N. Craswell, H. Zaragoza, and S.E. Robertson, "Microsoft Cambridge at TREC-14: Enterprise Track," *Proc. 14th Text REtrieval Conf. (TREC-14),* 2005.
[17] K. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman, "Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster," *Proc. Second Int'l Conf. Human Language Technology (HLT '02),* 2002.
[18] R. Krovetz, "Viewing Morphology as an Inference Process," *Proc. 16th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '93),* 1993.
[19] F. Diaz, "Personal Communication," 2007.
[20] E.M. Voorhees and D. Harman, "Overview of TREC-9," *Proc. Ninth Text REtrieval Conf. (TREC-9),* 2001.
[21] J.P. Marques de Sá, *Applied Statistics.* Springer Verlag, 2003.
[22] J.D. Lafferty and C. Zhai, "Document Language Models, Query Models, and Risk Minimization for Information Retrieval," *Proc. 24th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '01),* 2001.
[23] G. Mishne, "Using Blog Properties to Improve Retrieval," *Proc. First Int'l Conf. Weblogs and Social Media (ICWSM '07),* 2007.
[24] W. Kraaij, T. Westerveld, and D. Hiemstra, "The Importance of Prior Probabilities for Entry Page Search," *Proc. 25th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '02),* 2002.
[25] A. Berger and R. Miller, "Just-in-Time Language Modelling," *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICASSP '98),* 1998.
[26] D.M. Blei and J.D. Lafferty, "Dynamic Topic Models," *Proc. 23rd Int'l Conf. Machine Learning (ICML '06),* 2006.
[27] K.-Y. Chen, L. Luesukprasert, and S.-C.T. Chou, "Hot Topic Extraction Based on Timeline Analysis and Multidimensional Sentence Modeling," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 8, pp. 1016-1025, Aug. 2007.
[28] R.C. Swan and J. Allan, "Automatic Generation of Overview Timelines," *Proc. 23rd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '00),* 2000.
[29] H.L. Chieu and Y.K. Lee, "Query Based Event Extraction along a Timeline," *Proc. 27th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '04),* 2004.
[30] Y. Yang, A. Lad, N. Lao, A. Harpale, B. Kisiel, and M. Rogati, "Utility-Based Information Distillation over Temporally Sequenced Documents," *Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07),* 2007.
[31] F. Diaz and R. Jones, "Using Temporal Profiles of Queries for Precision Prediction," *Proc. 27th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '04),* 2004.
[32] I. Mani, J. Pustejovsky, and R. Gaizauskas, *The Language of Time: A Reader.* Oxford Univ. Press, 2005.

17. We thank one of the anonymous reviewers for the suggestion of this research direction.

Wisam Dakka received the PhD degree in computer science from Columbia University before joining Google. He has been a software engineer at the search quality team at Google since 2008. His research interests lie in the area of databases and information retrieval, with a special interest in faceted search.

Luis Gravano received the BS degree from the Escuela Superior Latinoamericana de Informática (ESLAI), Argentina, in 1991, and the MS degree from Stanford University in 1994. He received the PhD degree in computer science from Stanford University in 1997. He has been on the faculty of the Computer Science Department, Columbia University, since 1997, where he has been an associate professor since 2002. In 2001, he was a senior research scientist at Google (on leave from Columbia University).

Panagiotis G. Ipeirotis received the BSc degree from the Computer Engineering and Informatics Department (CEID) of the University of Patras, Greece, in 1999, and the PhD degree in computer science from Columbia University in 2004. He is an associate professor at the Department of Information, Operations, and Management Sciences at Leonard N. Stern School of Business of New York University. His area of expertise is databases and data mining, with an emphasis on management of textual data. He is a member of the IEEE.