Temporal Profiles of Queries

ROSIE JONES
Yahoo! Research
and
FERNANDO DIAZ
University of Massachusetts, Amherst

Documents with timestamps, such as email and news, can be placed along a timeline. The timeline for a set of documents returned in response to a query gives an indication of how documents relevant to that query are distributed in time. Examining the timeline of a query result set allows us to characterize both how temporally dependent the topic is, as well as how relevant the results are likely to be. We outline characteristic patterns in query result set timelines, and show experimentally that we can automatically classify documents into these classes. We also show that properties of the query result set timeline can help predict the mean average precision of a query. These results show that meta-features associated with a query can be combined with text retrieval techniques to improve our understanding and treatment of text search on documents with timestamps.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation*

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Time, temporal profiles, ambiguity, precision prediction, query classification, event detection, language models

ACM Reference Format:

Jones, R. and Diaz, F. 2007. Temporal profiles of queries. ACM Trans. Inf. Syst. 25, 3, Article 14 (July 2007), 31 pages. DOI = 10.1145/1247715.1247720 http://doi.acm.org/10.1145/1247715.1247720

1. INTRODUCTION

We have accesss to many collections of time-stamped documents. For example, email messages have a header giving the time and date they were sent and new articles typically come annotated with the date they were written or published.

This research was carried out while F. Diaz was interning at Overture Research (now Yahoo Research).

Authors' addresses: R. Jones, Yahoo! Research, 3333 Empire Ave., Burbank, CA 91504; email: jonesr@yahoo-inc.com; F. Diaz, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts, Amherst, MA 01003; email: fdiaz@cs.umass.edu.

Despite the existence of such collections, traditional information retrieval systems often do not exploit temporal information. By remaining temporally agnostic, strictly content-based systems fail to recognize subtopic or ambiguous structures in the query results, which may be reflected in the temporal distribution of documents that the query retrieves. Consider the example of the query "iraq war" submitted to a news corpus. A strictly content-based system will return a ranked list with documents discussing the 1991 and 2003 conflicts, without distinguishing them. Ideally, we would like the system to detect that two temporal clusters of documents exist in the retrieved set of documents. Temporally biased queries occur frequently in both standard collections and the on web [Li and Croft 2003; Diaz and Jones 2004].

If we can understand the temporal behavior of a query, we can automate the decision of whether to elicit relevance feedback, or modify an information retrieval system in other ways. We may also be able to use the temporal properties of the query result-set to diagnose the quality of the retrieval.

We propose distinguishing three temporal classes of queries. The first query type describes queries which are *atemporal*, taking place at any time. For these queries, the ideal relevance metric would be content based. The second query type describes queries which are *temporally unambiguous*, taking place at a specific period in time. For these queries, the ideal relevance metric would combine content and temporal information. The third query type describes those which are *temporally ambiguous*: taking place during one of several possible episodes. For this query type the ideal relevance metric would involve identifying the episodes, finding characteristic, or earliest documents within each episode, and possibly eliciting relevance feedback over the temporal domain.

In addition, we investigate the role of time when the system is ignorant of the true class or temporal distribution of a query. To this end, we consider the task of *precision prediction*. A key missing component in information retrieval systems is self-diagnostic tests to establish whether the current system can provide reasonable results for a given query on a document collection. In a language modeling retrieval system, content-based methods exist for predicting system performance given a query [Cronen-Townsend et al. 2002]. We expand on this work by adding time to the content features. We show that adding temporal features to a regression increases the predictive power.

This article is structured as follows: We begin in Section 2 by describing a probabilistic framework for reasoning about the temporal dimension of a query. These representations can be considered as the temporal analogs of query language models. We have found it helpful to extend temporal profiles by examining certain features of them. Our feature set is described in Section 3. Our dataset is described in Section 4. Given our feature-based representation, we can begin to classify queries into temporal types, to help identify those which would be good candidates for interactive disambiguation. We do this by first defining a set of temporal query classes in Section 5. In Section 6, we describe how we hand-labeled queries with the temporal classes *atemporal*, *temporally ambiguous* and *temporally unambiguous*. Then, using the features described in Section 3, we demonstrate the successful classification of queries in Section 7.

In Section 8, we describe the regression and classification algorithms we use for precision prediction. In Section 8.4, we show that we can improve the prediction of a query's average precision by taking into account temporal features. We conclude by describing a system for eliciting user feedback for temporally summarizing and disambiguating queries in Section 9.

The precision prediction experiments in this article expand results reported in Diaz and Jones [2004]. The temporal classification experiments (Section 7) and timeline visualizations (Section 9) have not previously been published.

2. TEMPORAL PROFILES

Our goal is to model the period of time relevant to a given query. For example, for the query "elections" our model would ideally give greatest weight to the days on which the most important elections occur, and smaller but non-zero weight to the days preceding elections, when political speeches may be made, and following elections, when votes may be tallied.

We have a document collection with time-stamps at our disposal. We use news stories from TREC corpora. Each document is annotated with a time-stamp as part of its header, corresponding to the date the document was published. We use the creation date of the document as a proxy for the date of the events referred to in the document. This is a reasonable assumption for news stories. When the time-stamps are unavailable, or for other types of text data, it may be useful to analyze the text in the document for date references [Mani and Wilson 2000]. For example, the content of the documents may contain text referring to relative dates such as "yesterday" or "next month", or absolute dates such as "September 11th, 2001". Using these dates we could refine our estimate of the day the document is most relevant. For this work, however, we restricted our attention to the date of publication of the document.

Given the document collection with time-stamps, one way to build the model would be to count documents containing the query words, assigning weight to each day on the basis of that count. In Figure 1, we see a simple frequency-based model for the queries "elections" and "iraq war" in a corpus of documents from 1986 through 2004.[1]

We notice several things about this model. First, for a day with no documents containing a query word, the frequency is zero. However, no documents may appear on a given topic, due to holidays, other news events gaining precedence, or vagaries about publication times near midnight, or across different timezones. It may still be reasonable to assume that a day is relevant to a query, if adjacent days are relevant to the query. For this reason we smooth across adjacent days, as we will describe in more detail in Section 2.3. Secondly, this frequency-based model gives weight to days with documents containing the term, regardless of how relevant those documents are. For example, we may suspect that documents in 1991 and 2003 are more relevant to the query "iraq war" than documents containing one or more of the terms in the mid-90s, but the document frequency model does not capture this. In Section 2.1, we describe

[1]This corpus collects documents from Tipster disks 1-5 (LDC93T3A), the English Gigaword (LDC2003T05), and HARD 2004(LDC2005T28).

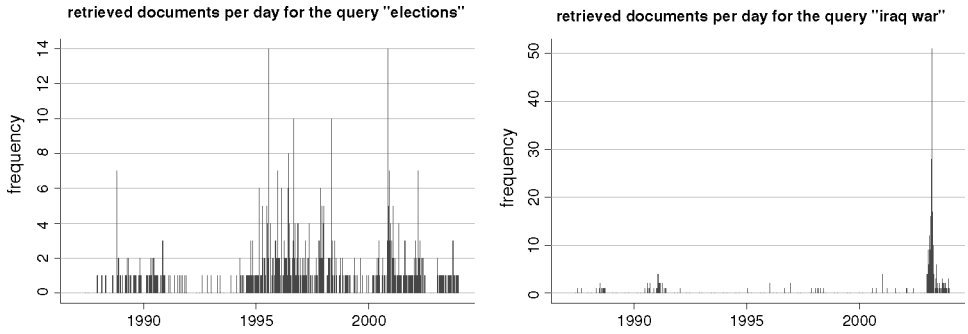


Fig. 1. Number of documents per day, for the queries "elections" and "iraq war" in a collection consisting of documents from 1986 through 2004. Histograms generated using the top 1000 retrieved documents.

a way of incorporating document relevance into the model. Finally, we should take into account the overall rate of documents occurring on a given day. We do this by smoothing with the background model, as described in Section 2.2.

2.1 Document Relevance Based Timeline

In a language modeling retrieval context [Croft and Lafferty 2003], we rank the documents in the collection according to their likelihood of having generated the query:

$$P(Q|D) = \prod_{w \in V} P(w|D)^{q_w}. \tag{1}$$

Here, Q is the current query, D is the current document, V is the entire vocabulary of words in the collection and q_w is the number of times the word w occurs in the query. A thorough review the language modeling approach can be found in other literature [Croft and Lafferty 2003].

We are interested in describing the *temporal* nature of a query using a probability distribution over days. We refer to this distribution as a *temporal profile* of the query. Formally, we would like to estimate the distribution $P(t|Q)$ where t is the day relevant to the searcher. Since this information is not present in the query, we encode the probability of the searcher selecting a specific date if the search engine provided such a capability.

We adopt a *relevance modeling* solution to this estimation problem [Lavrenko and Croft 2001]. That is, we want to look at the temporal information each of the top N documents provide and weight this information according to the document's probability of relevance, $P(Q|D)$. Thus we look at the first R documents retrieved, as a proxy for the set of relevant documents, and weight each by the estimated relevance. A schematic of this approach is shown in Figure 2. The formula for retrieving the top R documents was given in Formula 1.

Our temporal query model is initially defined as

$$\tilde{P}(t|Q) = \sum_{D \in R} \tilde{P}(t|D) \frac{P(Q|D)}{\sum_{D' \in R} P(Q|D')} \tag{2}$$

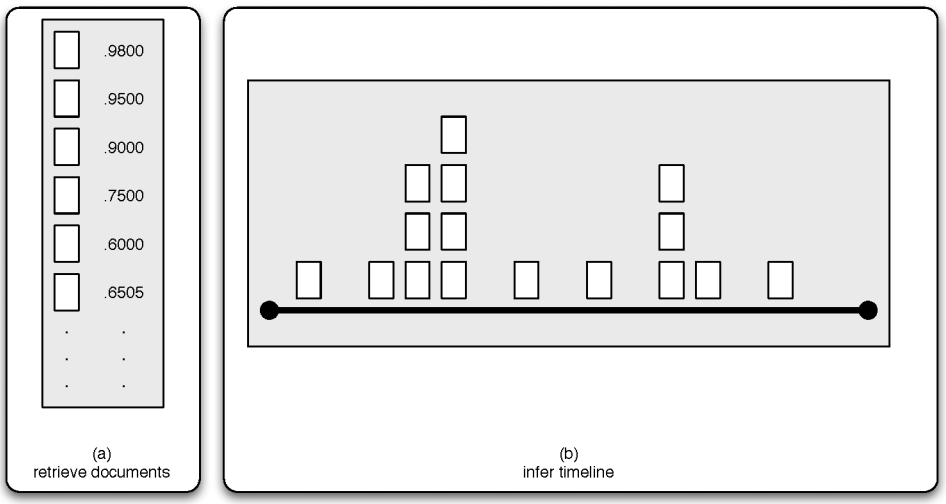


Fig. 2. We retrieve the R top-ranked documents for a query q, then place them along a timeline according to the timestamp, to generate the initial temporal for q.

where R is the set of top N documents. Our granularity is on the day scale; that is, the value of t is a single day. The first factor in this summation represents the temporal information we have about the document. We represent the temporal information of the document as a distribution over dates. Recalling $P(Q|D)$ is the document retrieval score, we note that the second factor in this summation is merely the normalized retrieval score.

In our experiments, the temporal information about a document is extracted from the document timestamp. Because each document contains a unique timestamp, our model reduces to a dirac delta on the day of timestamp,

$$\tilde{P}(t|D) = \begin{cases} 1 & \text{if } t \text{ is equal to the document date, } t_D \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In Figure 3(a) we see the timelines built using document relevance. We see that for the query "iraq war" the documents in 1991 appear to be more relevant than the documents in the mid-1990s, leading to greater probability mass in that part of the temporal profile.

2.2 Smoothing by Using Background Temporal Model

The overall distribution of documents in our collection, the *background model*, provides useful information about the general characteristics of term frequency and document frequency over dates, independent of the query we are modelling. Using this background model we can improve our improve our probability estimates using *background smoothing*. Background smoothing plays two roles. First, background smoothing handles potential irregularities in the collection distribution over time. For example, certain dates may have a large number of articles compared to others. Second, background smoothing replaces zero probability events with a very small probability, allowing us to assign a very small

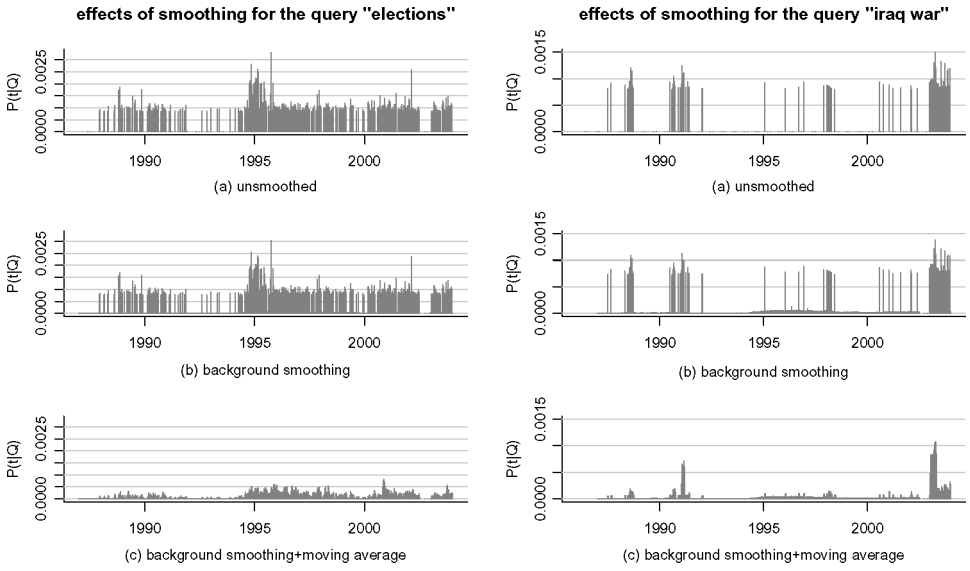


Fig. 3. Probability of relevance of each day, for the queries "elections" (left) and "iraq war" (right) on documents from 1986 through 2004.

likelihood of a topic being discussed on days where we have no explicit evidence. We use the distribution of the collection over time as a background model. This collection temporal model is defined by

$$\tilde{P}(t|C) = \frac{1}{|C|} \sum_{D \in C} \tilde{P}(t|D), \tag{4}$$

where C is the set of all documents in the collection.

Our estimate can then be linearly interpolated with this reference model such that

$$P'(t|Q) = \lambda \tilde{P}(t|Q) + (1 - \lambda)\tilde{P}(t|C) \tag{5}$$

where λ is a smoothing parameter that we set to 0.9 after initial experiments. We see in Figure 3(b) that background smoothing effectively adds non-zero mass to all days in the distribution.

2.3 Smoothing Across Adjacent Days

Since our model is discrete at the level of a single day, and news stories on a single topic may occur over a period of several days, we smooth our estimate of the model for a single day with the model for adjacent days. These kinds of smoothing techniques have been explored in the field of time series analysis. We use simple moving average smoothing. The smoothed estimate for a particular day is defined according to the previous p days,

$$P(t|Q) = \frac{1}{\phi} \sum_{i=0}^{\phi-1} P'(t - i|Q). \tag{6}$$

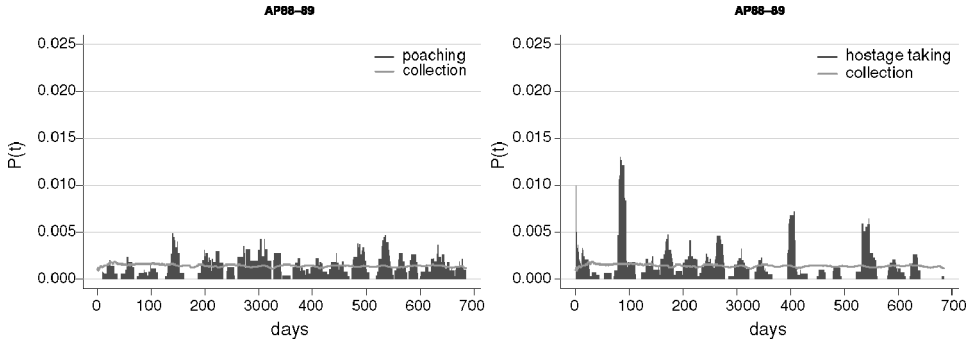


Fig. 4. Comparing the profiles for "poaching" and "hostage taking" we see that "hostage taking" appears to be relatively more episodic. For reference, we also show the profile generated from the entire document collection.

In our experiments, the period, ϕ, is always 14, smoothing the probability for a day with the 14 preceding days, but not subsequent days. This type of retrospective smoothing is common in fields such as market analysis where only historic information is available. Improvements could be made by smoothing with days both before and after the reference day or preferentially weighting the reference day. In general, the selection of ϕ should depend on the temporal granularity of the topic. For example, some topics will span only a few days while others span several months. We assume that, for the most part, the topics we consider occur at a granularity consistent with our smoothing parameter.

The distribution $P(t|Q)$ in Eq. 6 is our final estimate of the temporal profile. In Figure 3(c), we see the final smoothed models for the queries "election" and "iraq war".

3. FEATURES OF TEMPORAL PROFILES

In Section 2, we described the estimation of temporal profiles. Figure 4 depicts the temporal profiles for the queries "poaching" and "hostage taking" over the AP88-89 corpus. The profile for "hostage taking" appears relatively more episodic, with each episode presumably corresponding to a single real-world hostage taking event. "Poaching" appears relatively more uniform. In this section, we will define a set of features for discriminating between temporal profiles. These features will be used both for characterizing the temporal type of a query's profile, in Section 7, as well to predict the precision of queries, in Section 8.4. While not exhaustive, this set of features captures, in our opinion, the most important temporal aspects of temporal profiles.

Each feature has a different numerical range. After generating the features, we normalize them by shifting the minimum to zero and scaling by the range, to give features which lie between zero and one. In the descriptions of features, we will refer both to the raw feature values, and these normalized values.

3.1 Kullback–Leibler Divergence from the Collection Distribution

In language model based information retrieval, a query's "clarity" is rank-correlated with the effectiveness of the query at retrieving a precise topic

[Cronen-Townsend et al. 2002]. This content clarity measure assumes that the distribution of words in documents retrieved for a good query will be distinct from the background distribution. The clarity measure is defined as the Kullback–Leibler (KL) divergence between the query language model $P(w|Q)$ and the collection language model. Formally, the clarity score is defined as,

$$D_{\mathrm{KL}}(P(w|Q), P(w|C)) = \sum_{w \in V} P(w|Q) \log \left(\frac{P(w|Q)}{P(w|C)} \right). \tag{7}$$

A larger KL divergence indicates a clearer query. We will refer to this clarity measure as *content clarity*.

We propose an analog to content clarity for the temporal domain, by measuring the difference between the distribution over time of documents retrieved in response to a query, and the distribution over time of documents in the collection as a whole. This can be quantified by taking the KL divergence between the collection temporal model and the query temporal model. That is,

$$D_{\mathrm{KL}}(P(t|Q), P(t|C)) = \sum_{t=1}^{T} P(t|Q) \log \left(\frac{P(t|Q)}{P(t|C)} \right). \tag{8}$$

We will refer to this feature as temporal KL divergence, or *temporalKL*. The spiky nature of our example query, "hostage taking" (Figure 11) is clearly captured by this feature which has the value 0.57, which normalizes to 0.126 with respect to our collection. At the same time, the relatively a-temporal query, "poaching", exhibits a much lower KL divergence, with a KL of 0.247, which normalizes to 0.000 with respect to our collection.

Note that although temporalKL captures the deviation of documents retrieved for a query from the general distribution of documents over time, it may not allow us to distinguish between queries corresponding to events taking place at a single time, (such as "turkish earthquake 1999") and temporally ambiguous queries (such as "iraq war").

3.2 Autocorrelation

While the KL divergence gives us a test of similarity to the temporal background model ($P(t|C)$), it does not provide a measure of the randomness of the query time series. To test this, we use the first-order autocorrelation of the time series,

$$r_1 = \sum_{t=1}^{T-1} \frac{(P(t|Q) - \frac{1}{T})(P(t+1|Q) - \frac{1}{T})}{\sum_{t=1}^{T} (P(t|Q) - \frac{1}{T})^2}. \tag{9}$$

The autocorrelation of a uniform distribution is $r_1 = 0$. When queries contain a strong inter-day dependencies, the autocorrelation value will be high, suggesting a structure to the time series. For example, autocorrelation is high in cases where a high $P(t|Q)$ tends to predict a high $P(t+1|Q)$; likewise with low values. Such behavior indicates that there is predictability to the time series.

In Figure 11, the bursty episodes indicative of hostage events contribute to a higher autocorrelation of 0.938, which normalizes to 0.719 with respect to our collection. Similarly, the relative uniformity of the "poaching" query leads to a

smaller autocorrelation of 0.921, which normalizes to 0.611 with respect to our collection.

3.3 Statistics of the Rank Order of $P(t|Q)$

Another way to capture the dynamics of the time series is to consider how much of the probability distribution is contained in the peaks, and how much in the low-probability regions. In order to do this, we can measure the *kurtosis* of the time series, which is high for distributions with a sharp peak, and lower for distributions with less extreme values. In order to focus on the peaks as a single concept, we consider the rank order of the time series. In these cases, we reorder the days in decreasing $P(t|Q)$. We then construct a distribution over *ranks* where, for rank ρ, $P(\rho|Q)$ is defined as the value of $P(t|Q)$ for the day at that rank. We now consider statistical properties of $P(\rho|Q)$. Specifically, we look at the kurtosis of the rank order. The kurtosis is defined by,

$$\begin{aligned} \text{kurtosis} &= \frac{\mu_4}{\mu_2^2} \\ &= \frac{\sum P(\rho|Q)(\rho_i - \mu)^4}{[\sum P(\rho|Q)(\rho_i - \mu)^2]^2}, \end{aligned} \tag{10}$$

where μ_i is the ith central moment and μ is the mean. The kurtosis measures the "peakedness" of the curve.

As with temporalKL, the peaked nature of Figure 11 is represented in this feature. However, in this case, we inspect the *rank ordered distribution*, providing us a measure based on a smoothed representation of the peaks.

3.4 Burst Model

An alternative measure for temporal structure follows from Kleinberg's burst model [Kleinberg 2002]. Because several new features will be derived from this model, we present a brief overview of the relevant aspects. The burst model assumes that a hidden state machine generates some number of documents each day.[2] The actual number of documents produced is dependent on the state of the machine on that particular day. In our case, we would assume that there are two states: idle and event. We present a diagram of this model in Figure 5. In the idle state, documents are produced in batches so that, if the machine never leaves the event state, it generates documents uniformly across the time span. In the event state, the machine generates significantly more documents than in the idle state. These generation probabilities are analogous to a hidden Markov model's output probabilities. We hand-tuned the model parameters to a scaling parameter of $s = 2$ and the state-change parameter $\gamma = 1.1$.

A second important aspect of the burst model deals with the probability of transition between states. In our situation, this probability can be viewed as the "momentum" for the machine staying in a state. In Figure 5, such probabilities would be associated with the arrows. These probabilities are analogous to a hidden Markov model's transition probabilities.

[2]In particular, we use Kleinberg's $\mathcal{B}_s^2$ automaton.

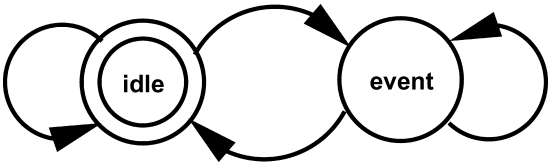


Fig. 5. Two-state model used for generating burst features. The idle state generates documents in a manner consistent with a uniform distribution over time. The event state generates many more documents per day than the idle state. Further parameters control the probability of moving between the states. This property controls the smoothness of the transitions between states. The burst model can be considered as a hand-tuned hidden Markov model.

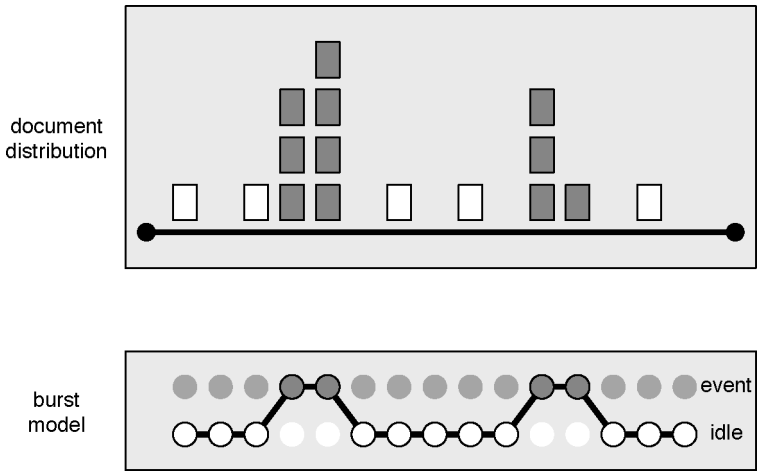


Fig. 6. The top frame shows the distribution of retrieved documents over the time span of the collection (one rectangle=one document). Documents highlighted in red provide evidence of an event or "burst". The bottom frame shows a model of the burstiness of the distribution. The burst model considers each day in the time span as either being relevant to a burst (such a day is in the "event" state) or not relevant to a burst (such a day is in the "idle" state). The highlighted sequence of states represents a model of the transition between event and idle states over the course of the time span.

We should note here that we are dealing with the original document sample (the top N retrieved documents) used to estimate the temporal profile, not any estimated model, $P(t|Q)$, described in Section 2. That is, we look at the actual number of documents occurring on a day as opposed to a probability.

As mentioned, the burst model can be compared to a hidden Markov model. The major difference is that the burst model specifies all of the parameters usually learned in a hidden Markov model (i.e., there is no "learning" in the burst model). While we could certainly imagine training an HMM for our task, we were interested in gross features of a potentially inaccurate model as opposed to finding the true event and idle state decoding.

We are interested in the state transition sequence most likely responsible for the document distribution. As with HMMs, we can use dynamic programming to efficiently calculate the values for all such sequences. We consider the sequence with the highest value to be the most likely sequence. Figure 6 shows a decoding for a toy example.

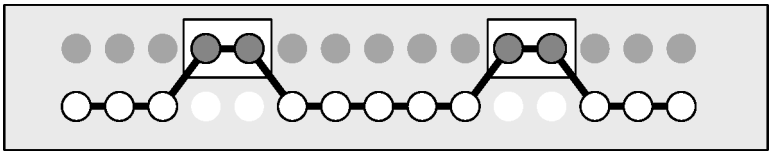


Fig. 7. Number of episodes. This feature of the burst decoding counts the number of sequences of days that the model predicts are in the burst state. In this example, we detect 2 episodes.

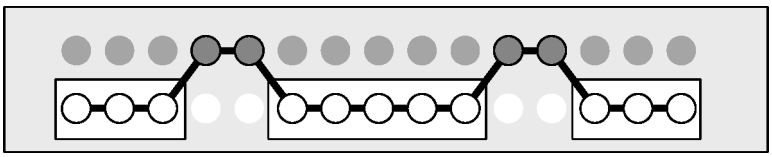


Fig. 8. Average time in the idle state.This feature of the burst decoding average number of days that the decoding is in the idle state before entering the event state. In this example, we detect three sequences of idle state days of lengths 3, 5 and 3. On average, then, we spend 3.67 days in the idle state.

Since we hypothesize that the eventfulness of a topic will be a good feature, we consider three measure of temporal structure from the decoding sequence. The first feature merely counts the number of episodes in the decoding. We refer to a sequence where the machine is only in the event state as an episode. A graphical depiction of this feature is presented in Figure 7. The second feature measure the average amount of time in the idle state. That is, we take the all of the sequences where the machine is only in the idle state and compute the average length. The average length of time the machine is in the idle state gives a measure of the overall significance of the topic over the time span in the collection. A graphical depiction of this feature is presented in Figure 8. The third feature measures the quality of the decoding by inspecting how much we prefer the given decoding over one which does not include transitions into the event state. That is, we compare the given decoding to one in which there are no transitions into the event state. Kleinberg refers to this as the weight of a burst. We expect the burst weight to show the "intensity" of the time profile when relevant documents are found. This may reflect queries corresponding to high-intensity situations which are distinct from the background model. A graphical depiction of this feature is presented in Figure 9.

The burst model for the profile for "poaching" (Figure 10) spends most of the time in the idle state and has few transitions into the event state. Meanwhile, the model for "hostage taking" (Figure 11) transitions 5 times and spends, on average, half as much time in the idle state. Combined with the intensity measure, these features point to a more temporally structured query.

4. DATA

We worked with two types of data in this work: news articles and web search query logs. In this section we describe these datasets.

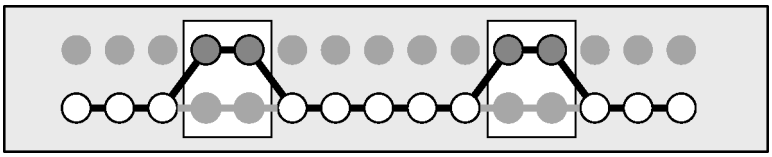
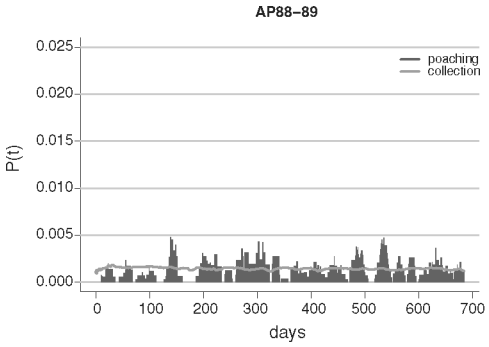
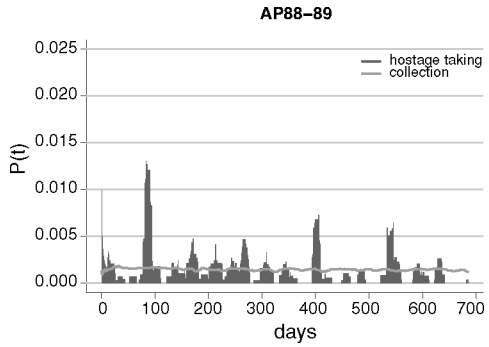


Fig. 9. Burst weight savings. For each sequence of days predicted to be in the event state, we compare the confidence in the prediction of an event sequence to the confidence in the prediction of an idle sequence. The differences between these confidences are then averaged to obtain our feature value. In this example, we have two event sequences whose confidence we will compare to an idle sequence for the same days (depicted in grey).



feature	value	value'
temporalKL	0.247	0.000
autocorrelation	0.921	0.611
kurtosis	2.476	0.000
burst weight savings	5.313	0.001
average amount of time in idle state	341.0	0.993
number of transitions into event state	2	0.000
content clarity	1.796	0.451
average precision	*0.565*	*0.565*

Fig. 10. Temporal profile and raw and normalized feature values for the query "Poaching" over the AP88-89 collection. We normalized feature values by shifting and scaling them to lie between zero and one. This query had the minimum score for both temporal KL and number of transitions into the event state, and so the scores for both of these features are zero after normalization.



feature	value	value'
temporalKL	0.570	0.126
autocorrelation	0.938	0.719
kurtosis	3.921	0.124
burst weight savings	8.465	0.025
average amount of time in idle state	132.6	0.225
number of transitions into event state	5	0.429
content clarity	1.422	0.265
average precision	*0.139*	*0.139*

Fig. 11. Temporal profile and raw and normalized feature values for the query "Hostage Taking" over the AP88-89 collection. We normalized feature values by shifting and scaling them to lie between zero and one.

4.1 TREC Corpora

We use standard TREC corpora for document-level experiments [Voorhees and Harman 2001]. The TREC corpora often consist of newswire articles containing time and date stamps. However, these standard collections often contain properties undesirable for our experiments. For examples, there are often large subcollections of documents missing temporal information. Others have highly

nonuniform temporal distributions of documents. While these issues suggest interesting research questions, we decided to focus on temporally uniform collection with no missing temporal data.

We chose three news collections for our experiments. We used the Associated Press collection from Tipster disks 1 and 2 (AP), the Wall Street Journal Collection also from Tipster disks 1 and 2 (WSJ), and the combined New York Times, Associated Press, Xinhua News Service collections from the AQUAINT disk (AQUAINT). Unless otherwise noted, the Lemur language modeling toolkit was used for text retrieval [Allan et al. 2003]. Query likelihood ranking was performed and document models were smoothed using Jelinek–Mercer smoothing ($\lambda = 0.6$). All documents and queries were stopped using the SMART stopword list and stemmed using the Krovetz stemmer [Salton 1971; Krovetz 1993]. Dates were extracted from document identifiers.

Both the AP and WSJ collections have roughly 100 TREC queries associated with them. Because the queries were constructed with respect to the larger collections, we often found queries with few relevant documents in our AP and WSJ collections. Therefore, we only used queries with more than 15 relevant documents in our collections.

4.2 Weblog Corpus

In addition to the TREC experiments, we were interested in the use of temporal profiles for classifying web search logs. In these experiments, we use a web search log from an ISP for March 2003–June 2003. The logs represent a time-stamped record of all of the search sessions by users during this period. Here, a session refers to a set of queries issued by a single user segmented when a gap of 10 minutes occurs between queries. For example, if a user issued the queries "salsa", then "salsa dancing", and finally "salsa new york" without 10 minute gap between queries, then the pseudo-document here is the concatenation of these strings date-stamped with the day on which they occurred. Because this corpus consisted of a huge number of very short documents, a simple relevance ranking based on Jaccard distance was used [Manning and Schütze 1999]. We manually constructed a set of queries was derived for the search log corpus, with 24 queries in each of the three classes. The queries were constructed without reference to the documents retrieved. These queries are publicly available.[3] All queries were stemmed and stopwords were removed.

5. TEMPORAL CLASSES

We construct the temporal profile of a query by examining the distribution of the documents it retrieves across the timespan of the corpus. Figure 11 shows the temporal profile for the query "hostage taking". We gave details of how the temporal profiles are constructed in Section 2. Note that the profile contains several spikes, which may correspond to bursts of documents appearing at particular times in the span of the collection. The fact that there are multiple bursts suggest that there may be multiple real-world events generating the documents, which means there is some ambiguity in the query about which real-world event

[3]http://www.cs.cmu.edu/∼rosie/data/2006TOIS/.

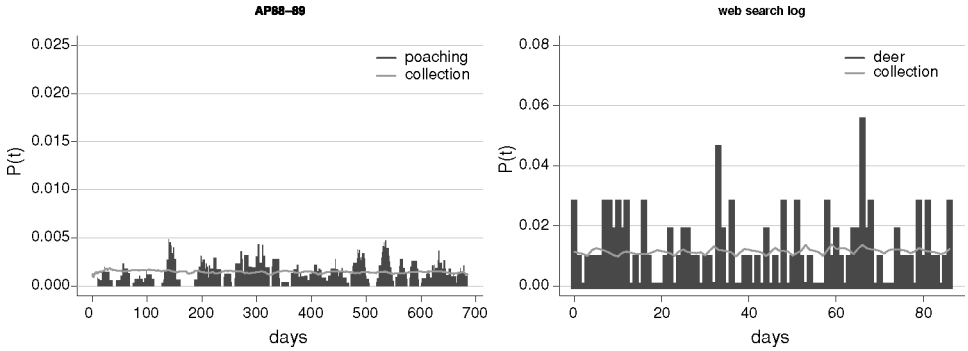


Fig. 12. Temporal profiles of two atemporal queries in two collections. On the left, the temporal profile of the query "poaching" (TREC Topic 77) estimated from the AP 1988 and 1989 collection. On the right, the temporal profile of the query "deer" estimated from a web search session log for March 2003–June 2003. Notice how the retrieved documents are more or less evenly distributed across the collection timespan. In both graphs, the background collection distribution is plotted as a reference.

is referred to. In this section, we define three temporal classes of a query by distinguishing three common patterns of document profiles retrieved in response to queries. We describe these classes below.

5.1 Atemporal Queries

Atemporal queries are relatively time-invariant with respect to the document collection. For example, the query "opinions about the death penalty" focuses on a topic which is not sensitive to time. Atemporal queries correspond to a topic which is ongoing. While the details of documents relevant to the query may change over time, we expect their distribution in time to be similar to the overall distribution of documents. When overall document volume increases, we expect the volume of documents relevant to atemporal queries to increase too. Figure 12 shows the time-relevance profile of two queries, "poaching" and "deer", in two collections. These are topics of perennial interest, and thus no structure can be seen in the profiles.

5.2 Temporally Unambiguous Queries

Temporally unambiguous queries are relatively distinct with respect to the time dimension. For example, the query "turkish earthquake 1999" refers to a specific span in time. Figure 13 shows the profiles for two queries, "earthquake in armenia" and "matrix". This profile suggests that the query refers to a specific point or period in time. Note that a query is only temporally unambiguous with respect to a specific collection. For example, if our web search log were extended both forward and backward in time, several peaks in interest would exist (placing in into our third class). Similar behavior would exist if there were another earthquake in Armenia in our collection. In addition, these examples demonstrate the difference between unanticipated events such as natural disasters and anticipated events such as a movie release. While not investigated in this work, detecting these subclasses also might be useful.

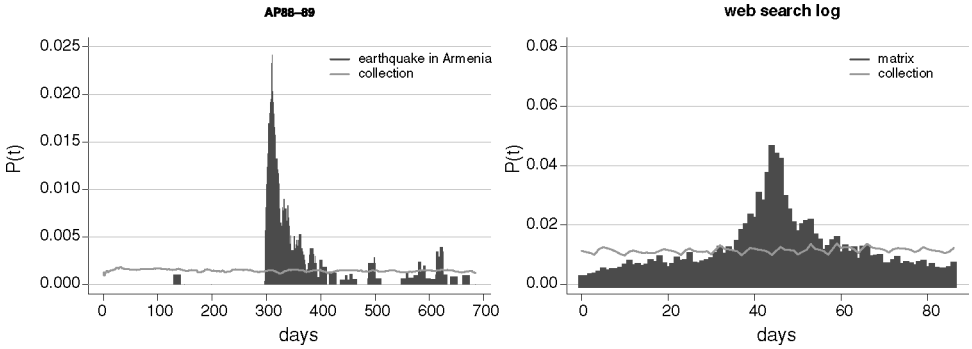


Fig. 13. Temporal profiles of two temporally unambiguous queries, "earthquake in armenia" and "matrix". In both cases, the retrieved set of documents seem to refer to a single event in the collection. The temporal profile on the left also shows the behavior or an unexpected event (a natural disaster). On the other hand, the temporal profile on the right depicts an anticipated event (a movie release).

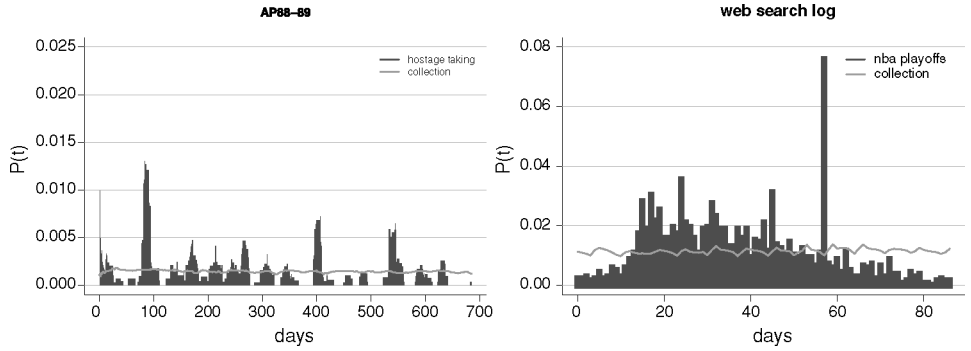


Fig. 14. Temporal profiles of two temporally ambiguous queries, "hostage taking" and "nba basketball playoffs". The behavior of the profiles in both cases lies somewhere between the temporally unambiguous and atemporal queries. While the temporally unambiguous queries contain a unique peak, these profiles consist of multiple, shorter peaks. The existence of these peaks represents possible sub-events of the query's topic not found in atemporal profiles.

5.3 Temporally Ambiguous Queries

Lastly, *temporally ambiguous* queries refer to the combination of several events and hence might be considered ambiguous if the user is looking for information about a specific event. For example, the query "iraq war" may refer to either the conflict in 1992 or the conflict in 2003. Figure 14 shows the profiles of two queries, "hostage taking" and "NBA basketball playoffs". Here, the profiles have a less distinct temporal profile than the temporally unambiguous queries but are nonetheless distributed quite differently from the temporal profile of the entire collection. An interesting example not shown here is the subclass of periodic queries.

6. MANUAL CLASSIFICATION OF QUERIES INTO TEMPORAL CLASSES

In this section, we describe how we constructed labeled training and test sets for the temporal classes described in Section 5. We also show how these classes correlate with average precision where we have relevance judgments available.

6.1 Manual Classification of TREC ad-hoc Queries

Attempting to work with a standard set of queries, we used the TREC *ad hoc* query sets. Since none of the queries in these sets were annotated with temporal classes, they had to be hand-classified. We classified more than 50 queries using the TREC topic descriptions. Annotators were not provided temporal profiles. Specifically, annotators were asked to judge, based on the topic, description, and narrative fields, whether a query was requesting multiple events, a single event, or had no preference. The annotators labeled 18 common queries, and agreed on 13/18 of the common queries, for 72% agreement. On inspecting the documents retrieved for the queries, we were better able to discern events underlying the queries, and relabeled the queries on which the annotators disagreed. This suggests that the topic, description and narrative field may not be sufficient to identify how time-dependent a query is. A system for automatically identifying when a query is temporally ambiguous may therefore be useful since users may not be able to make this judgment themselves without inspecting documents.

Interestingly, we found that TREC *ad hoc* queries belonged only to the atemporal and temporally ambiguous classes; that is, queries either did not refer to a specific event or referred to a set of events. Each of the AP and WSJ collections had 51 associated queries (though not exactly the same 51 queries, since we limited queries to those with 15 or more relevant documents in the given collection). In the AP collection, 28 of these queries were atemporal; the remaining 23 were temporally ambiguous. In the WSJ collection, 34 of these were atemporal; the remaining 17 were temporally ambiguous. Examples of classifications are the query "Airbus Subsidies" (TREC query 51) which was manually classified as temporally ambiguous, and the query "U.S. Economics" (TREC query 57) which was manually classified as atemporal.

This data provides us with a set of TREC queries for which relevance judgements are available, classified into our temporal classes. We will examine the average precision of these queries in Section 6.4. We give results of classifying into these two classes in Section 7.2.

6.2 Augmented Query Set with Manually Constructed Temporally Unambiguous Queries for the TREC Collection

In Section 5, we described three temporal classes, but we found only two of them in the TREC *ad hoc* queries. We augmented the set of annotated TREC queries with a collection of new, temporally unambiguous queries. This augmenting set was generated from a list of natural disasters, deaths, and other single-event queries from the time period of the corpora. An example of one of the temporally unambiguous queries we added is "Dalai Lama wins Nobel Peace Prize". The addition of this query set gives us queries for all three classes defined over the TREC ad hoc collection. We do not have relevance judgments for the third

Table I. Average and Standard Deviation of Average Precision, Broken Down by Query Temporal Class

	AP			WSJ		
Class	n	$\bar{x}$	s	n	$\bar{x}$	s
temporally ambiguous	23	0.31	0.25	17	0.32	0.37
atemporal	28	0.29	0.38	34	0.27	0.41

In our datasets, atemporal queries have lower average precision on average, and a greater standard deviation in their average precision. This may be because the temporally ambiguous queries we used deliberately sought documents from a range of time periods.

class, temporally unambiguous queries, but we can perform retrieval with these queries, and use the resulting profiles for three-way classification experiments. We report results for classifying into these three classes in Section 7.3.

6.3 Novelty Queries Classified by Temporal Type

Another source of TREC queries is based on the AQUAINT news corpus which covers the years 1996 through 2000. The TREC 2003 Novelty track developed a set of 50 queries for this corpus which were classified as either "opinion" or "event" queries by NIST labelers. The definitions used for these types were:

—*Event* topics are about a particular event that occurred within the time period of the collection. Relevant information pertains specifically to the event.

—*Opinion* topics are about different opinions and points of view on an issue. Relevant information takes the form of opinions on the issue reported or expressed in the articles.

On inspection, the classes "opinion" and "event" corresponded well with our classes "atemporal" and "temporally unambiguous", respectively.[4]

We performed two-way classification experiments with these queries as well. We report results classifying into these two classes with novelty queries in Section 7.1.

6.4 Average Precision across Temporal Classes

Table I shows average precision scores, averaged across the queries in the two temporal classes for which we had relevance judgments: temporally ambiguous and atemporal queries. We find that in our datasets, atemporal queries have lower average precision on average, and a greater standard deviation in their average precision. This may be because the temporally ambiguous queries deliberately sought documents from a range of time periods. We would expect quite different results for relevance judgements from users unaware of the temporal ambiguity in their query, and expecting documents referring to a single incident.

[4]Since our definitions did not take into account the time period of the collection, the TREC Novelty event queries are a subset of our "temporally unambiguous" queries.

7. AUTOMATIC CLASSIFICATION OF QUERIES INTO TEMPORAL TYPES

In this section, we describe experiments on automatic classification of queries into the three classes *atemporal*, *temporally unambiguous*, and *temporally ambiguous*, on the basis of the time profiles of retrieved documents. The general approach we employ here is to use supervised machine learning. We take a sample of pre-classified queries, use a machine learning algorithm to learn a rule that can be used to classify them, then apply that rule on a set of held-out queries, and measure the rule's accuracy.

We used the machine learning algorithm of decision tree induction, implemented by the Weka Toolkit [Witten and Frank 1999], to learn a classifier and to perform classification. The decision tree induction algorithm, introduced by Quinlan [1993] greedily chooses features to add to the tree, based on their discriminative power on the training sample. When no more discriminative questions can be found, the majority class of the examples satisfying all of the conditions is given, for each branching path in the tree. When using the learned decision tree to classify new examples, the classification program starts from the base of the tree, and check whether the test example satisfies the stated condition. Each answer leads to a different branch in the tree, while when we reach a leaf of the tree we assign the class that is represented there.

There are two baseline systems. The first baseline system classifies queries with the majority class in the training set. That is, if the majority of the training set queries were of a particular class, this baseline would always predict that class. The second baseline attempts to learn a classification based only on the content clarity [Cronen-Townsend et al. 2002] of the query. Content clarity experimentally is able to distinguish between *topically* unambiguous and ambiguous queries. Nevertheless, content clarity provides a surprisingly strong baseline. Because content clarity is also one of our features, this baseline should demonstrate the contribution of the content clarity feature in the composite system. Note that for the web search-log corpus, content clarity does not have a clear analog because we are not using a language model retrieval system. Therefore, it was removed from our feature set for these experiments.

In order to address small query sets, all our classification experiments use ten-fold stratified cross-validation.

7.1 Results Automatically Classifying Novelty Queries: Temporally Unambiguous and Atemporal

In our first task, we look at distinguishing between temporally unambiguous and atemporal queries. These are the extrema of the temporal query types we are modeling, and distinguishing between these two classes may be the easiest of the problems we wish to address. The TREC Novelty queries match these two types, and have a corresponding corpus of relevance-labeled documents, as described in Section 6.3. The majority baseline is 56%, since 56% (28 out of 50) of the novelty queries are in the class *temporally ambiguous*. The remaining 22 or 44% are in the class *atemporal*.

Table II shows the accuracy of the trained decision tree compared to the baselines. With accuracy of 62%, content clarity gives a 10% relative improvement

Table II. Accuracy of A Decision Tree Trained to Distinguish *Atemporal* From *Temporally Unambiguous* Queries, Using the TREC Novelty Collection

	Accuracy
majority class	.56
content clarity	.62
temporal features	.68
temporal features+content clarity	.70

Table III. Accuracy of A Decision Tree Trained to Distinguish *Atemporal* from *Temporally Ambiguous* Queries. Experiments were Performed Separately on the TREC AP and WSJ Collections

	AP	WSJ
majority class	.54	.66
content clarity	.54	.66
temporal features	.71	.68
temporal features+content clarity	.73	.73

over the majority-class baseline. It seems that content clarity provides some indication of the temporal ambiguity of a query. Without temporal information, it is surprising that content clarity says anything about distinguishing these query classes. It is clear, though, that there are important aspects not measured in the feature. Using temporal features alone, we get a relative improvement of 21% over the majority baseline and 9% relative improvement using content clarity aline. This improvement over content clarity suggests that temporal features better capture aspects of the topics. Content and temporal features can be exploited simultaneously to give an accuracy of 70%, a 25% relative improvement over the majority baseline.

7.2 Ad hoc Queries: Temporally Ambiguous and Atemporal

As described in Section 8, TREC *ad hoc* queries naturally fell into the two classes *temporally ambiguous* and *atemporal*. Table III shows results for the automatic classification experiments. We see that while it is always the case that temporal features are necessary to differentiate between these two classes, it is less clear that they are sufficient; the improvement over the baseline is slight for WSJ. Inspecting the queries falsely classified as temporally ambiguous, we noticed queries whose temporal characteristics were more subtle. For example, queries such as "reform of the us welfare system" or "what backing does the national rifle association have" were misclassified. These are less a collection of events than manifestations of popular or political interest. Looking at the temporal profiles of these queries confirms this. The temporal profiles of these queries rise and fall in the news according to popular or political opinion. We believe that event-driven topics are better candidates for disambiguation than topics whose temporal characteristics are indirectly the result of popular opinion. This raises an interesting area of future work addressing the distinction between these types of topics.

Table IV. Accuracy for Decisions Trees Trained to Distinguish Between All Classes on Both TREC Collections and a Web Search Log

	AP	WSJ	Search Log
majority class	.38	.37	.27
content clarity	.40	.45	—
temporal features	.65	.45	.75
temporal features+content clarity	.65	.43	—

The set of queries for the TREC collections are augmented by queries constructed to be temporally unambiguous.

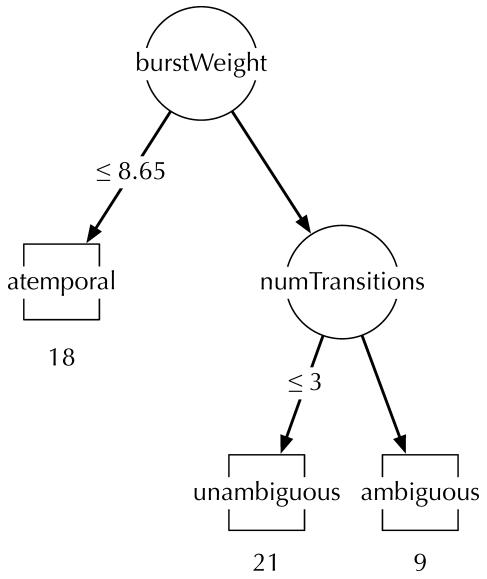


Fig. 15. Decision tree learned to classify queries in the web search log dataset (Table IV). Numbers under the leaf nodes represent number of class instances correctly labeled. Each class has 24 instances. Most of the temporally unambiguous and atemporal queries are correctly classified using only features of the burst model (burst weight and number of state transitions).

7.3 Temporally Unambiguous, Temporally Ambiguous, and Atemporal Queries

For our final classification experiments, we turn to the complete task of distinguishing between all three temporal categories of queries. We performed these experiments over the augmented TREC *ad hoc* queries, and the web log queries, using the labels we described in Section 6.

We see in Table IV that we can make substantial improvements from the majority-class baseline for all collections. What is interesting is that much better performance was achieved on the web search log data. There are several explanations for this behavior. First, web searching behavior for the selected temporally unambiguous queries is dramatically divergent. This is apparent when we inspect the decision tree learned for the web search log set (Figure 15). Almost all of the temporally unambiguous queries are correctly classified by the conjunction of very large burst path cost savings (weight) and having fewer than

two state transitions. The type of profile exhibiting this behavior will have one high peak. Second, the web search behavior for the selected atemporal queries is dramatically uniform. Again, inspecting the decision tree, we notice that a large majority of the atemporal queries are correctly classified by low burst path cost savings. It is the significantly better performance in these two classes that explains the better overall performance. The performance on temporally ambiguous queries is comparable with the other corpora.

Thus we find that we are able to classify queries into the three temporal classes with much better accuracy than the baseline. This gives us a way of triaging queries for different kinds of treatment. For example, we may wish to provide a timeline for disambiguation for temporally ambiguous queries, while for temporally unambiguous queries we may like to highlight the date of the event peak.

This kind of triage will be most useful when we can also identify which queries have high-quality or low-quality results using our default retrieval methods. In the next section, we will examine predicting a query's precision using content clarity and the temporal features of the retrieved set.

8. PREDICTING A QUERY'S AVERAGE PRECISION

Cronen-Townsend et al. [2002] showed that content clarity correlates with average precision, when using the Spearman rank-correlation test. This allows an information retrieval system to rank a set of queries by the likely quality of results. Low quality queries can be improved by further processing by requesting feedback from the user. However, clarity does not permit the system to predict the likely precision of any individual query, beyond a binary classification. In this section, we analyze the Spearman rank correlation of temporal features with average precision. We also build models to predict the average precision of queries using a combination of temporal and content features, that is, using the temporal features we described in Section 3 along with content clarity as a feature describing the content. While any relationship between these features and average precision may be nonlinear, we first perform linear regression. This will allow us to compare the importance of features by examining their coefficients. We then use our measures of temporal clarity along with content clarity as input features to a neural network for predicting average precision.

For all experiments described in this section, we normalize the input features to lie between zero and one, by shifting and scaling the values. We now define a training and test collection, as well as performance measures. For these retrieval experiments, we use the TREC collection of news documents and queries described in Section 4.

8.1 Spearman Rank Correlation

The Spearman rank correlation coefficient allows us to examine the relationship between predictor and predicted variables, without assuming any particular structure to that relationship. For example, with the Spearman rank correlation coefficient, we can measure whether increases in content clarity lead to increases in average precision, without assuming that these are, for example,

Table V. Spearman Rank Correlation Coefficient for Correlation with Average Precision, for Content Clarity, and Our Proposed Temporal Features, for AP and WSJ. Statistically Significant Rank-Correlations are Shown in Bold Along with Their Significance Levels

	AP		WSJ	
Feature	R^2	Prob R^2	R^2	Prob R^2
autocorrelation	**0.36**	**0.01**	−0.05	0.72
burstAverageIdleTime	0.10	0.50	−0.02	0.87
burstWeightSavings	**0.28**	**0.05**	0.05	0.73
contentClarity	**0.54**	**5.0e-05**	**0.50**	**2.0e-04**
kurtosis	0.14	0.32	0.22	0.12
temporalKL	0.01	0.94	0.14	0.31

linear increases. Positive correlation using the Spearman rank correlation test tells us that there is a relationship between the variables. However, it does not tell us how to predict one variable from the other. The Spearman rank correlation is also not defined over multiple variables simultaneously. Thus, we cannot use it to find whether we can improve our understanding of the average precision of a query by combining predictor variables.

8.2 Linear Regression

Linear regression allows us to combine multiple predictor variables, to predict *linear* changes in the average precision. That is, we are finding a linear relationship between our predictor variables and average precision. The coefficients of the variables show us their relative importance in predicting average precision, though two variables which are correlated may wind up with lower coefficients. For this reason, we will also show the coefficients for each variable when used in isolation for predicting average precision. Since linear regression looks for linear relationships between variables, it is a stronger test with strong assumptions about the relationship between variables. We are less likely to find statistically significant correlations with linear regression than with Spearman rank correlation, particularly when the underlying relationship is nonlinear.

8.3 Neural Networks

Neural networks allow us to model nonlinear relationships between combinations of predictor variables. The hidden layers allow the representation of subcombinations of features, which may aid with prediction. A neural network outputs a prediction of average precision for any input, and we can then compare the prediction with the actual average precision for a query. We measure the difference between actual and predicted average precision.

8.4 Results of Spearman Rank Correlation of Features with Average Precision

Table V shows the Spearman rank correlation with average precision for each feature in isolation. We see that the correlation of content clarity with average precision is much higher than all other features, and that this correlation is statistically significant. This reproduces the results obtained by Cronen-Townsend

Table VI. Correlation from Linear Regression: Average Precision is the Dependent Variable

	AP		WSJ	
Independent Variables	Train	Test	Train	Test
content clarity	0.33	0.21	0.41	0.36
temporal features	0.40	0.15	0.38	0.17
content + temporal features	**0.71**	**0.52**	**0.75**	**0.60**

Independent variables are content clarity, and our measures of temporal clarity. Test correlation was found by cross-validation, by fitting the line using training data, then measuring correlation with held-out test data. For the row labeled "content + temporal features" we use all our temporal features, as well as content clarity as inputs to the linear regression.

et al. [2002]. For our temporal features, the correlation is much lower, and for most features the measure of correlation is not statistically significant. This means that most of the temporal features are not predictive of average query precision, when used in isolation. However, note that a combination of these features may be predictive of average query precision. For the AP dataset, autocorrelation was positively correlated with the rank of average query precision at the 0.01 level, and burst weight savings was correlated at the 0.05 level. That means these two features may contribute to an improved predictive model of average precision, when combined with each other and content clarity, if they are not redundantly correlated with one another.

8.5 Results of Linear Regression of Features against Average Precision

Table VI shows the correlation using linear regression lines between average precision and measures of query clarity. Test correlation was found by cross-validation, by fitting the line using training data, then measuring correlation with held-out test data. Note that for both AP and WSJ, the combination of content and temporal measures shows a much stronger correlation with average precision than content clarity or temporal features alone. This means that our measures of temporal clarity contribute to the understanding of the likely effectiveness of a query with respect to a corpus. Note also that the correlation scores remain high when tested using cross-validation.

The variables with strong predictive power are shown in Table VII with their coefficients for predicting average precision. The coefficients shown are with ridge regression, which performs normalization and removes potentially irrelevant features. This led to the best predictive results. To estimate the standard deviation of the coefficients, we performed pairs bootstrapping for 10,000 iterations, without normalization. Thus, these standard deviations give an upper bound on the uncertainty of the estimates of coefficient strength. Note that while some of the features are unstable, we obtain high correlation scores, even on a held-out set of data not used for fitting the regression lines.

Interestingly, the magnitude of the coefficient for content clarity is not the largest. We find that temporalKL has a negative coefficient. This shows that queries with temporal profiles very different from the background model are likely to have low average precision. This suggests that queries which retrieve documents from an unusual subset of days in the collection are likely to perform

Table VII. Coefficients of Individual Features in Linear Ridge Regression for Predicting Average Precision for AP and WSJ Data. Also Shown is the Sample Standard Deviation Over 10,000 Iterations of Bootstrapping Pairs Linear Regression, Without Normalization

	AP		WSJ	
Feature	Coeff	s^2	Coeff	s^2
autocorrelation	0.43	0.22	−0.24	0.19
burstAverageIdleTime	—	0.19	0.70	6.58
burstNumTransitions	0.19	0.19	1.0	8.23
contentClarity	0.97	0.34	0.96	0.13
kurtosis	0.28	0.31	—	0.21
temporalKL	−1.3	0.45	−0.83	0.26

poorly. These may be good candidates for a relevance feedback interface that highlights the days on which retrieved documents appeared, and allows the user to select the appropriate timeframe. We discuss a possible interface of this form in Section 9.

How do we explain the predictive performance of *temporalKL*, which does not predict average precision when used in isolation, but which has a negative coefficient in conjunction with the other features? We can infer that it explains some parts of average precision which are not explained by the other features, but only when we know the values of the other features.

8.6 Results of Neural Networks to Predict Average Precision

Neural networks can be used to learn nonlinear functions. We used the Weka implementation of neural networks [Witten and Frank 1999]. Our target function is average precision. In all cases, we used one hidden layer. In the case of content clarity only as an input feature, we used a single node in the hidden layer. In all other cases we used three nodes in the single hidden layer. All input features are connected to all nodes in the hidden layer. We used a learning rate of 0.3 and momentum of 0.2, 10% validation set and 500 training epochs, with early termination of training if the accuracy on the validation set grew worse over 20 epochs. We used 80% of the data for training, 10% for validation, and 10% for testing.

As a baseline, we guessed the mean average precision over all queries. The mean average precision was 0.28 over all queries on WSJ, and 0.30 over all queries on AP. When we use this value for average precision for every query in the test set, we can calculate how much each query deviates from this level of average precision, and calculate root-mean squared error (error in terms of difference between predicted and actual query average precision), and root relative squared error (error in terms of a percentage of the actual average precision of a query). With these baseline average precision values, predicted average precision was on average 0.20 from the true value (root mean squared error), with 100% root relative squared error, as shown in the first row of Table VIII. By using the other features as input to a neural network, we hope to gain predictions of average precision for each query which are more reliable than guessing these baselines.

Table VIII. Error in Predicting Average Precision Using a Neural net with 1 Hidden Layer

	AP		WSJ	
Features	RMSE	RRSE	RMSE	RRSE
baseline	0.20	100%	0.22	100%
NN: content clarity	0.53	101%	0.18	87%
NN: temporal features	0.31	98%	0.27	131%
NN: content + temporal features	**0.27**	**88%**	**0.13**	**63%**
LR: content + temporal features	**0.23**	**92%**	**0.18**	**82%**

RMSE is root mean squared error of the predicted value on a held-out test set. RRSE is root relative squared error on the held-out test set. In the final row we show these measures on the model built with linear regression. We see that neural networks performed better at prediction than linear regression, so we gain predictive power from the nonlinear transformations.

We see in Table VIII that content clarity in isolation reduced the root relative squared error for WSJ to 87%, and that temporal features in isolation do not reduce the root relative squared error. However, the final row of Table VIII shows that using the combination of temporal and content clarity with a neural net, we can reduce both the root mean squared error and the root relative squared error from the default. This means that we are able to predict average precision with greater accuracy than the default, and greater accuracy than using content clarity alone. In the final row, we show these measures on the model built with linear regression. We see that neural networks performed better at prediction than linear regression, so we gain predictive power from the nonlinear transformations.

8.7 Summary of Precision Prediction

In this section, we showed that temporal features can aid in predicting the average precision of a query's retrieval results. This means that we identify poorly performing query and elicit further processing, such as relevance feedback and temporal disambiguation.

9. VISUALIZATION

In this section, we move on to the question: How should we treat temporally ambiguous queries through user interaction? If we wish to perform query-specific relevance feedback based on the temporal properties of a query, we need a way of showing the user the distribution of relevant documents over time. The visualization technique we will describe here is applicable to all types of queries, but most useful for temporally ambiguous queries.

We will assume that the component events of a query are nonoverlapping so we only need to consider segmenting the information into an annotated timeline. This task was divided into three parts: determine the time-spans corresponding to events, construct a language model for each event, and use the event language models to build event summaries.

9.1 Detecting Events

Recall that Kleinberg's burst model, described in Section 3.4, models the rate of document production for the documents retrieved for a query. This is modeled

as a combination of two rates: a low rate of document production, in which documents are generated from the *idle* state, and a higher rate of document production, when documents are generated from the *event* state.

The time-spans of an event can then be extracted from the state sequence decoding of the burst model. We combine unbroken chains of event states into a single event. For example, if January 3, 1992 and January 4, 1992 both are both predicted to be in the "event" state, then we treat both the dates as days in the same event.

9.2 Constructing Event Language Models

Event language models can easily be constructed by using the documents which lie within the time-spans as evidence for the event language model. So, if the burst model decoding indicated that an event occurred during a particular two-week range, the subset of the original top N documents retrieved for the query, which lie within this time-span, would be used to make the event language model. Formally,

$$\hat{P}(w|E) = \frac{1}{|E|} \sum_{D \in E} P(w|D), \tag{11}$$

where E is the subset of top N documents within the particular timespan. Equation (11) calculates the maximum likelihood estimate for this distribution. Subsequent references to $P(w|E)$ represent the maximum likelihood model smoothed using Jelinek–Mercer smoothing with a weightof 0.2 for the background model.

9.3 Building Event Summaries

Given these event language models, then, we can think of two ways of presenting summaries of their information to the user. First, we can inspect the distribution of terms in $P(w|E)$. Presenting a list of terms in order of decreasing probability might result in many corpus-wide high-probability terms being displayed. Therefore, we look at the pointwise KL divergence measure of per-term contribution to Eq. 7 [Tomokiyo and Hurst 2003]. Specifically, the pointwise KL divergence is defined as,

$$\delta(p||q) = p(w) \log \frac{p(w)}{q(w)}, \tag{12}$$

where p is our event model and q is some reference distribution over words. This gives us a measure of the distinguishing quality of each word in the event model with respect to the reference model. Using the collection language model as our reference distribution may result in a good summary of the query as a whole but may not serve to distinguish between the events in the topic. Therefore, we use the query model, $P(w|Q)$, as our reference model q; this model is estimated out of the top N documents as,

$$P(w|Q) = \sum_{D \in R} P(w|D) \frac{P(Q|D)}{\sum_{D' \in R} P(Q|D')}, \tag{13}$$

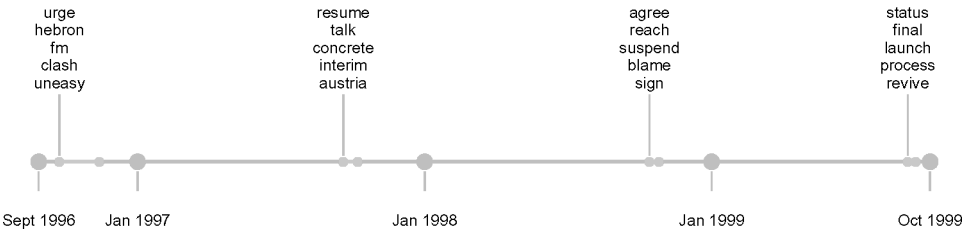


Fig. 16. Non-extracted Summaries: The AQUAINT corpus was used to visualize the temporal profile of the query "Israel–Palestine". Kleinberg's burst model is used to detect event boundaries. Event language models are built from documents within these ranges. Summaries consist of the terms with the highest point Kullback–Leibler divergence with respect to the collection language model.

where R is the set of top N documents and there is a uniform prior over the documents.Therefore, the terms in our event model are ranked according to $\delta(P(w|E)||P(w|Q))$. We refer to the top five terms from this list as the *nonextractive summary*.

Figure 16 shows the nonextractive summary for the query "Israel–Palestine". We notice that the words chosen hint at the nature of the events, though we cannot tell for example, who is blaming who. Increasing the size of the extracted terms from unigrams to bigrams or trigrams could lead to more informative extracted summaries.

Another way to generate summaries from the event models is to consider sentences from the set of in-event documents. Let *sentence likelihood* refer to the sentence level analog of document likelihood such that,

$$P(s|E) = \prod_{w \in V} P(w|E)^{s_w}, \tag{14}$$

where s_w refers to the number of times word, w, occurs in the sentence. If $\mathcal{S}$ is the set of all sentences in these documents, we can calculate the likelihood of each sentence and pick the most likely sentence. Formally, the event summary is the sentence, s^*, with the highest likelihood,

$$\begin{aligned} s^* &= \operatorname*{argmax}_{s \in \mathcal{S}} P(s|E) \\ &= \operatorname*{argmax}_{s \in \mathcal{S}} \prod_{w \in V} P(w|E)^{s_w}. \end{aligned} \tag{15}$$

We are interested in relatively high precision and useful summaries so we restrict $\mathcal{S}$ to the set of document titles. We refer to the most likely title as the *extractive summary* of an event. Figure 17 shows the extractive summary for the query "Israel–Palestine". We notice that a sentence makes the summary easier to interpret. However, we cannot be sure that some of the properties of the event are missing from the single sentence chosen.

10. RELATED WORK

In terms of classifying queries into temporal classes, Swan and Jensen [2000] performed retrieval with 10 queries from the TDT2 corpus. They perform

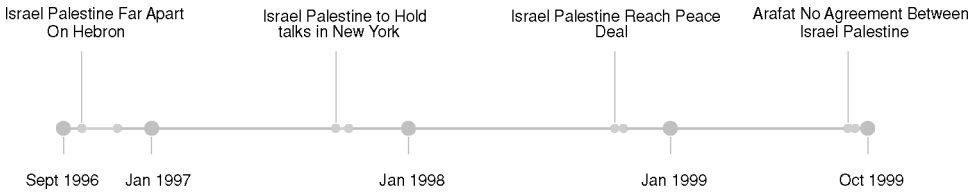


Fig. 17. Extracted Summaries: Using the same query and event models as in Figure 16, summaries are the in-event document titles with the highest likelihood of having been produced by the event language model.

feature discovery with the matching documents, and characterize the queries into types, based on the number of features found by TimeMines. They describe those queries for which TimeMines found many features as having many subtopics. They suggest that queries for which few or no features were discovered can be described as topics which are fairly static. Our work differs in that we categorized the temporal-category of queries before performing retrieval and feature extraction, and showed that we can use the features to predict those categories.

Li and Croft [2003] proposed the inclusion of temporal evidence into the document prior of a language modeling retrieval system. The authors first manually triaged TREC queries into temporal classes based on the distribution of known relevant documents. Queries in particular classes were then given different document priors. Their work demonstrates how to incorporate temporal information if we know which temporal class a query belongs to. However, the authors do not address the problem of automatically detecting which class queries belong to.

In terms of precision prediction, Cronen-Townsend et al. [2002] introduced content clarity as a content-based method for predicting system performance given a query. We expand on this work by adding time to the content features. He and Ounis [2004] investigated alternative content-based measures, and found that the retrieval step is not imperative, as statistics from the corpus as a whole can predict as well as those from a retrieved set. There has also been work on query-based event extraction [Chieu and Lee 2004].

In terms of visualization, Swan and Jensen [2000] extract noun-phrases and named entities from documents, and find correlations between phrases and timespans using a χ^2 test. They select the top phrases according to the χ^2 statistic, and show that these correlate with significant events in the news. They also present an interface for visualizing the top stories in a corpus based on the features discovered. This is essentially a data mining task, in that they find features only for a subset of the documents, and not for specified documents. We generalize this approach to the *ad hoc* retrieval scenario and conduct analysis with respect to the searcher's query.

Chieu and Lee [2004] evaluated sentence-based temporal summarization methods. In this work, temporal information is extracted from the natural language of the sentence and timelines are constructed based on sentence relevance and number of relevant sentences on that date. Timelines are then

evaluated according to user satisfaction. Our work focuses on evaluating temporal profiles for auxiliary tasks such as classification and prediction.

11. DISCUSSION

Temporal profiles provide a method for predicting relevant dates for a particular query. Learning algorithms can successfully exploit information encoded in these profiles to improve tasks such precision prediction and query triage. Despite these positive results, we believe there are several areas left to explore.

We constructed and evaluated temporal profiles using collections where documents were uniformly distributed over time. In practice, collections often contain nonuniform document distributions. Future work should be cautious about applying our estimation technique without first testing the uniformity of the document distribution. Adapting the estimation technique for nonuniform distribution remains an open problem.

We found that the quality of retrieval results is correlated with the distribution in time of the documents retrieved. Since we found that we can identify temporally ambiguous queries, these are good candidates for disambiguation with a small time series interface. We proposed a candidate interface in Section 9. We propose that a small time series be shown both to summarize the results, and to allow the user to provide temporal relevance feedback if desired. Users frequently select spelling suggestions and related query options in search engines, since these assist the searching process [Anick 2003]. Adding a temporal summary and feedback mechanism when the time component is significant should lead to increased satisfaction of information needs.

The definition of the temporal classes were guided by our intuition about what types of queries would be helpful for disambiguation or pseudo-feedback. One way to evaluate these categories would be to simulate the effects of these decisions. For example, if our classifier predicted a temporally unambiguous query, we might prefer documents in that time period without feedback. In the case of temporally ambiguous queries, we might request disambiguation using the interface from Section 9.

In order to understand the human issues with temporal feedback, we need to conduct user studies. We can do this in two ways. With small groups of users in laboratory settings, we can evaluate task completion and survey users about their level of satisfaction. We can also evaluate it with large groups of users by implementing and deploying the feedback mechanism in a search engine. We can then measure click-through-rate and session length. If the interface appears useful to web searchers, we may see click-through on the page which is higher than in a control group. If the feedback is effective, we may see greater interaction with results, with lower rates of search abandonment, and longer dwell times on result pages. We expect the temporal interface to be a good complement to similar, content-based disambiguation interfaces such as document and terminological feedback.

Finally, our temporal profiles use document time stamps to estimate $P(t|D)$. Oftentimes document also contain a variety of temporal information in the

content. When automatically detected, these dates can provide additional evidence for use when estimating our temporal profiles.

12. CONCLUSIONS

We have demonstrated that temporal information can successfully be leveraged for several retrieval tasks. For example, we showed that the temporal estimation procedure outlined in Section 2 generated a representation useful for precision prediction, temporal classification, and visualization. In all cases, temporal information improves performance over merely looking at the topical aspects of documents.

More importantly, our work presents an instance of a more general class of metadata ambiguity problems. That is, in addition to the temporal domain, this work can be extended into other metadata such as geography, language, and familiarity. As metadata are increasingly attached to documents, the incorporation of metadata into content-based systems will become an important research area for the information retrieval community.

REFERENCES

ALLAN, J., CALLAN, J., COLLINS-THOMPSON, K., CROFT, B., FENG, F., FISHER, D., LAFFERTY, J., LARKEY, L., TRUONG, T. N., OGILVIE, P., SI, L., STROHMAN, T., TURTLE, H., AND ZHAI, C. 2003. The lemur toolkit for language modeling and information retrieval. http://www-2.cs.cmu.edu/~lemur/.

ANICK, P. 2003. Using terminological feedback for web search refinement: A log-based study. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM, New York, 88–95.

CHIEU, H. L. AND LEE, Y. K. 2004. Query based event extraction along a timeline. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)* (Sheffield, UK) ACM, New York, 425–432.

CROFT, W. B. AND LAFFERTY, J. 2003. *Language Modeling for Information Retrieval*. Kluwer Academic Publishers.

CRONEN-TOWNSEND, S., ZHOU, Y., AND CROFT, W. B. 2002. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*. ACM, New York, 299–306.

DIAZ, F. AND JONES, R. 2004. Using temporal profiles of queries for precision prediction. In *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*. ACM, New York, 18–24.

HE, B. AND OUNIS, I. 2004. Inferring query performance using pre-retrieval predictors. In *Proceedings of the 11th Symposium on String Processing and Information Retrieval (SPIRE 2004)* (Padova, Italy). Lecture Notes in Computer Science, Springer-Verlag, New York.

KLEINBERG, J. 2002. Bursty and hierarchical structure in streams. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*. ACM, New York, 91–101.

KROVETZ, R. 1993. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1993)*. ACM, New York, 191–203.

LAVRENKO, V. AND CROFT, W. B. 2001. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*. ACM, New York, 120–127.

LI, X. AND CROFT, W. B. 2003. Time-based language models. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management (CIKM 2003)*. ACM, New York, 469–475.

MANI, I. AND WILSON, G. 2000. Robust temporal processing of news. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, 69–76.

MANNING, C. D. AND SCHÜTZE, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

QUINLAN, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann, Franciso, CA.

SALTON, G. 1971. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ.

SWAN, R. AND JENSEN, D. 2000. TimeMines: Constructing timelines with statistical models of word usage. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000)*. ACM, New York, 73–80.

TOMOKIYO, T. AND HURST, M. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, D. M. Francis Bond, A. Korhonen, and A. Villavicencio, Eds. 33–40.

VOORHEES, E. M. AND HARMAN, D. K., EDS. 2001. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, Cambridge, MA.

WITTEN, I. H. AND FRANK, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan-Kaufmann, San Francisco, CA. http://www.cs.waikato.ac.nz/ml/weka/.

Received January 2006; revised September 2006; accepted January 2007