Towards Recency Ranking in Web Search

Anlei Dong Yi Chang Zhaohui Zheng Gilad Mishne
Jing Bai Ruiqiang Zhang Karolina Buchner Ciya Liao Fernando Diaz
Yahoo! Inc.
701 First Avenue, Sunnyvale, CA 94089
{anlei, yichang, zhaohui, gilad, jingbai, ruiqiang, karolina, ciyaliao, diazf}@yahoo-inc.com

ABSTRACT

In web search, *recency ranking* refers to ranking documents by relevance which takes freshness into account. In this paper, we propose a retrieval system which automatically detects and responds to recency sensitive queries. The system detects recency sensitive queries using a high precision classifier. The system responds to recency sensitive queries by using a machine learned ranking model trained for such queries. We use multiple recency features to provide temporal evidence which effectively represents document recency. Furthermore, we propose several training methodologies important for training recency sensitive rankers. Finally, we develop new evaluation metrics for recency sensitive queries. Our experiments demonstrate the efficacy of the proposed approaches.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services Web-based services

General Terms

Algorithms

Keywords

Recency ranking, recency sensitive query classification, temporal features, recency modeling

1. INTRODUCTION

When a user submits a query to a search engine, he/she expects to obtain relevant search results. Classic notions of relevance focus on topical relevance. Web search introduces non-topical facets to relevance. For example, incorporating the authoritativeness of the information source in ranking can significantly improve user satisfaction. In this paper, we will argue that the freshness of documents can and should influence ranking and evaluation.

A large number of new web pages are created every day, and existing web pages are outdated with high rates. If stale documents are presented to users, they may seriously degrade search experiences. In recent years, the temporal dimension of web search has been studied from different perspectives, such as web dynamics, crawling, temporal features, and modeling. Web page recency has been effectively taken into account for some specific applications (e.g. research publication databases and news article page ranking). However, there has not been a large scale, comprehensive study focused recency sensitive evaluation and ranking.

In web search, *recency ranking* refers to ranking documents by relevance which takes freshness into account. Recency ranking on the large-scaled web search offers several unique and challenging research problems. First, ranking algorithms for recency sensitive queries need to satisfy both topical relevance and freshness. In order to ensure robustness, the system needs to promote recent content *only when appropriate*, so as to avoid degrading performance for other queries classes. Second, recency sensitive queries operate at different time scales. The freshness of a document depends on the time sensitivity of the query. For example, for the query "WSDM", related pages are time-sensitive to the year. However, for a breaking-news query, the relevance of the results displayed may be sensitive to days or even hours. Third, measuring the true age of a document is hard. In general, temporal features cannot be accurately extracted from web pages; usually, only weak temporal evidence can be obtained. Finally, gathering data for training and evaluation requires temporally sensitive judgments synchronized with retrieval runs.

We propose a system which addresses recency sensitive queries using a query classification framework. That is, our system can be broken into two models: a high-precision recency sensitive query detector and a specialized recency sensitive ranker. Training a specialized ranker allows the system to model the unique data distribution and features useful for recency ranking. The main contributions of this paper include:

1. the analysis and formulation of recency ranking problem,
2. the development of a breaking-news query classifier with high accuracy and reasonable coverage,
3. the extraction of document temporal evidence, and
4. the development of new algorithms and strategies for recency ranking models.

2. RELATED WORK

The works that are most related to our approach include [8] and [22], which also directly improve ranking recency in web search. However, our approach provides a more generic solution than previous works, in terms of the content and query perspective. Diaz [8] proposed a solution to integrate search results from a news vertical search into web search results, where the news intent is detected by either inspecting query dynamics or using click feedback; our

WSDM'10, February 4–6, 2010, New York City, New York, USA.

approach is to optimize ranking recency on all web pages directly, covering a more generic content perspective. Zhang *et al.* [22] proposed a ranking score adjustment method on year-qualified-queries, for which a few simple but effective adjustment rules are applied to the ranking results based on the timestamps extracted from the documents. Our approach can be applied on all time-sensitive queries, which covers a more generic query perspective.

The following prior works exploit the temporal dimension in general web search. Baeza-Yates *et al.* [2] studied the relation between the web dynamics, structure and page quality, and demonstrated that PageRank is biased against new pages. In T-Rank Light and T-Rank algorithms [3], both activity (i.e., update rates) and freshness (i.e., timestamps of most recent updates) of pages and links are taken into account for link analysis. Cho *et al.* [6] proposed a page quality ranking function in order to alleviate the problem of popularity-based ranking, and they used the derivatives of PageRank to forecast future PageRank values for new pages. Nunes *et al.* [17] proposed to improve web information retrieval in the temporal dimension by combining the temporal features extracted from both individual document and the whole web. Pandey *et al.* [18] studied the tradeoff between new page exploration and high-quality page exploitation, which is based on a ranking method to randomly promote some new pages so that they can accumulate links quickly.

Temporal dimension is also considered in other information retrieval applications. Del Corso *et al.* [7] proposed the ranking framework to model news article generation, topic clustering and story evolution over time, and this ranking algorithm takes publication time and linkage time into consideration as well as news source authority. Li *et al.* [15] proposed a TS-Rank algorithm, which considers page freshness in the stationary probability distribution of Markov Chains, since the dynamics of web pages is also important for ranking. This method is proved to be effective in the application of publication search. Pasca [19] used temporal expressions to improve question answering results for time-related questions. Answers are obtained by aggregating matching pieces of information and the temporal expressions they contain. Furthermore, Arikan *et al.* [1] incorporated temporal expressions into language model, and demonstrated experimental improvement in retrieval effectiveness.

Recency Query Classification plays an important role in Recency ranking. Diaz [8] determined the newsworthiness of a query by predicting the probability of a user clicks on the news display of a query. König *et al.* [14] estimated the click-through rate for dedicated news search result with a supervised model, which is to satisfy the requirement of adapting quickly to emerging news event.

3. RECENCY DATA ANALYSIS

In this paper, we target breaking-news queries, because breaking-news queries exhibit most typical issues with regards to recency in search ranking results and where it is clear that user experience can be improved by improving the ranking. Breaking-news queries include queries about topics which are in the news at the time when the query was entered. In other words, there was a "buzz" or major influx of content in the media on that topic at the time when the query was entered by the user. The event which the query refers to may have happened several days before the date of the query, but if there is still significant news coverage for that event the query should be classified as breaking-news. There are other non-breaking-news-queries which may also have recency problems. We first study breaking-news queries in attempt to solve the recency problem within this query category. With increased maturity of the technology, we will extend similar approaches to other queries.

Recency ranking data collection is different from regular ranking data collection. In regular ranking data collection, the relevance judgement for a query-url pair is usually static over time because document freshness does not affect the user satisfaction. The judgement for a recency-sensitive query-url pair, however, should incorporate the freshness of the document. For example, the epidemic *swine flu* was identified in April 2009. Therefore, for the query "swine flu" in April 2009, the web page of a news article reporting the identification of this epidemic can be appropriately labeled with the grade "excellent". A few days later, this web page becomes outdated as there have been many more web pages reporting the latest status of this epidemic, beyond its identification; thus, the grade should be demoted from "excellent".

We conducted an editorial test in order to confirm that there was indeed a relationship between objective recency and subjective recency. We sampled a set of queries automatically classified as recency-sensitive (details of this method are described in Section 4). From this set, editors selected only queries which were truly recency-sensitive for evaluation. For each query, we select the 20-30 top-ranked urls returned by a commercial search engine. We then employ different techniques to promote fresh urls. For example, based on the link discovery times, the fresher a page, the higher promotion score it should be given.

It is obvious that, to label recency data, a tuple $\langle \text{query}, \text{url}, t_{\text{query}} \rangle$ needs to be provided for judgement instead of only $\langle \text{query}, \text{url} \rangle$, where t_{query} is query issue time. If the judging time t_j of $\langle \text{query}, \text{url} \rangle$ is far behind query issue time t_{query}, it is impractical for the editors to do reliable judgement. Therefore, we collected sets of recency data periodically instead of collecting all the recency data for only one time. Each time we collected a set of recency data, we ask editors to judge the query-url pairs immediately, so that the query issue time and judging time are as close as possible. Collecting recency data periodically can also prevent the data distribution from being too biased towards a short period of time. For example, within one day, there are many similar queries related to the same breaking news, which are very different from the query distribution over a longer time span.

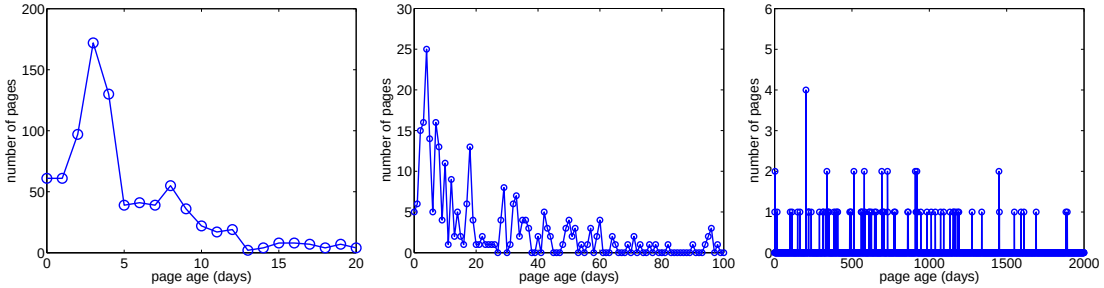
We then asked human editors for the following labels,

1. relevance judgments of query-url pairs
2. recency-sensitivities and true timestamps of a sample of urls

We apply five judgement grades on query-url pairs: perfect, excellent, good, fair and bad. For human editors to judge a query-url pair, we ask them to first grade it by non-temporal relevance, such as intent, usefulness, content, user interface design, domain authority, etc; then, the grade can be adjusted solely based on the recency of the result. More specifically, a result should receive a demotion if the date of the page, or age of the content, makes the result less relevant in comparison to more recent material or changes in time which alter the context of the query. This demotion should be reflected in the following judgment options:

- shallow demotion (1-grade demotion): if the result is somewhat outdated, it should be demoted by one grade (e.g., from excellent to good);
- deep demotion (2-grade demotion): if the result is totally outdated or totally useless, it should be demoted by two grades (e.g., from excellent to bad).

The advantages of this recency-demotion grading method include: 1) recency is incorporated into overall relevance so that the ranking model learning problem to be formulated as the optimization of a single objective function; 2) recency can also be decoupled



(a) no demotion (b) 1-grade demotion (c) 2-grade demotion

Figure 1: Page distribution with different ages at different recency demotion levels.

from overall relevance so that recency and non-recency relevance can be analyzed separately during learning.

A potential problem of the recency-demotion grading method is that shallow demotion and deep demotion may not demote time-sensitive documents to the most appropriate extent. It is possible that the recency demotion may be biased in the overall relevance judgement, i.e., the recency demotion can be either too aggressive for some cases or too conservative for some other cases. We carried out some heuristic studies and found that this judgement guideline is appropriate for most cases. Furthermore, such a recency guideline by one-grade or two-grade demotion is the most straightforward for human editors, whose judging needs to be efficient and can be somewhat error-prone.

The recency-sensitivity of a url means the page has a specific time aspect to it, which is related to news, recurrent events, blogs, buzz queries, etc; otherwise, it is a non-recency-sensitive page. Based on editorial judgements, 5,649 out of 16,327 pages, approximately 34.6%, are labeled as recency-sensitive pages. We further ask editors to label each recency-sensitive page with its ground-truth timestamp, which should be the date that the main content is created or published for each page. There are 1,888 recency-sensitive pages for which the editors can provide reliable dates.

Page age is an objective measurement of recency (which can computed using labeled date), while recency demotion is a subjective judgement by a human about recency. After conducting this editorial study, we found the following results. In the 1,888 recency-sensitive pages with labeled dates, 95 pages are demoted with two grades, 534 pages are demoted with one grade and 1,259 pages are not demoted. Figure 1 shows the page distributions with different ages at different recency demotion levels. We observe most of the non-demoted pages have ages within 10 days, most of the 1-grade-demoted pages have ages within 60 days, and the 2-grade-demoted pages have older ages. This is consistent with our expectation that non-demoted pages have younger ages than demoted pages overall. Note that in Figure 1 (a), the pages which are several days old have the highest frequencies, while there are fewer fresher pages such as those created within two days. This indicates that there are more recency queries that are related to the news that happen several days ago than those that happen within two days. At the same time, the search engine's crawler do not fully discover fresh contents, which is also a factor for this observation.

4. RECENCY SENSITIVE QUERY CLASSIFICATION

When a query is issued to the search engine, we use an automatic query classifier to determine if this query is a time-sensitive one or not. Not all queries require specialized handling to promote recent content; in fact, we show later that in some cases retrieval performance can actually degrade when our methods are applied to non-time-sensitive queries. Ideally, we would like to apply our approach only to the subset of breaking-news queries – those queries that are related to ongoing and upcoming events that are in the news. To achieve this we developed an automated classifier that first detects whether an incoming query is a breaking-news one; in this section, we describe this classifier.

The underlying assumption behind our breaking-news query identification is that when a topic is in the news, it tends to diverge from its regular temporal behavior: it becomes more prominent in the query stream, is mentioned more often in news articles, and so on. Existing methods for detecting such topics typically involve tracking the frequency of the topic over discrete time slots, and detecting irregularities such as a large divergence from the mean number of occurrences or a sudden burst of activity [13, 20].

In contrast, our classifier works not by modeling each individual topic and tracking it over time, but by modeling each discrete time slot, and comparing the models representing different time slots. The data we model in each time slot includes the queries users submitted during that period, as well as the news content published during it; the tool we use to model these is n-ngram language models. For each time slot, we collect all queries received by the search engine during that slot; we also collect a set of news articles published during that time slot. We then proceed by estimating two n-gram language models, $M_{C,t}$ and $M_{Q,t}$, representing the Content and the Query data at time t, respectively. To construct the langauge models, we consider each query as a sentence.

To identify whether a query q is a breaking-news one at time t, we would like to compare its affinity with the models representing t with its affinity to models representing time periods in the past. To this end, we first select several fixed time intervals, r_i: these will be the points in time to which we compare the current model. We choose these so that their relation to the current model would reveal an irregularity in the current model, if one exists. For example, typical historical time slots are the one starting 24 hours before t, the one starting one week earlier, or the one starting one month earlier. These intervals define several "reference" models – models representing the content and query data at times $t - r_i$. We refer to these models as $M_{C,t-r_i}$ and $M_{Q,t-r_i}$.

Next, we compute the generation likelihood of q according to the "current" query model, $P(q|M_{Q,t})$; for this, the model is smoothed using linear interpolation with a lower n-gram model. We repeat this computation, this time for each one of the reference models, obtaining several $P(q|M_{Q,t-r_i})$ values. Note that we use models representing identical or similar time spans (but starting and ending at different times) to minimize differences in the generation likelihood results stemming from model estimation parameters.

Finally, we compute the "buzziness" of the query according to the models built from the query stream as

$$\text{buzz}(q,t,Q) = \max_i \log P(q|M_{Q,t}) - \log P(q|M_{Q,t-r_i})$$

We repeat this computation for the Content models, obtaining a

Table 1: Using different reference models for query classification: incorporating past query data models, then content models, and an additional boost from the news shortcut signal.

Reference model	Precision	Coverage (% query traffic)
$M_{Q,t-r_i}$ $r_i = r_{prev\_day}$	0.72	1.39
$M_{Q,t-r_i}$ $r_i = r_{prev\_day}, r_{prev\_week}$	0.81	1.59
$M_{Q,t-r_i}$ $r_i = r_{prev\_day}$, $r_{prev\_week}, r_{prev\_month}$	0.78	1.64
$M_{Q,t-r_i}$ and $M_{C,t}$ $r_i = r_{prev\_day}$, $r_{prev\_week}, r_{prev\_month}$	0.81	1.81
$M_{Q,t-r_i}$ and $M_{C,t-r_i}$ $r_i = r_{prev\_day}$, $r_{prev\_week}, r_{prev\_month}$ with news shortcut boost	0.87	2.65

similar estimation of the buzziness, according to the content models, $\mathrm{buzz}(q,t,C)$. We then combine the two scores to obtain a final score,

$$\mathrm{buzz}(q,t) = \lambda \cdot \mathrm{buzz}(q,t,Q) + (1-\lambda) \cdot \mathrm{buzz}(q,t,C)$$

Building on previous work aimed at determining when to display a news shortcut next to the web search results [8], we also boost the buzz score of queries that triggered a news shortcut during those same reference time windows by a constant factor.

A query is considered as a breaking-news one if its final buzz score exceeds some threshold k.

To obtain optimal λ and k values, we collect manual judgments for a set of 2000 queries over several days. Within each day, judges were presented with queries submitted by search engine users on that day and asked to determine whether these queries refer to an upcoming, ongoing, or very recent event, for which web search users would prefer recent content. Queries were judged on the same day they were sampled. Judges were instructed that not every query that has an increase in volume is a breaking-news one: for example, following a death of an actor, there is typically some renowned interest in movies starring the actor; such queries (in contrast to queries about the actor himself) are not judged as breaking-news ones as the user is not likely to benefit from recent content. The λ and k values are chosen to maximize the accuracy of $\mathrm{buzz}(q,t)$ over this judged set.

In Table 1, we show the contributions of the various reference models used in our model, which incorporates content and query data for the reference models starting 24 hours earlier, one week earlier, and one month earlier than the current time t. The data in this table is aggregated over 600 queries sampled on several different days, and distinct from those queries using to optimize the model parameters. The accuracy of our classifier varies: during periods with a substantial amount of breaking-news events the precision increases to above 0.9, whereas quiet periods have lower precision. Depending on the amount of news in a given day, the coverage of our classifier—the number of queries classified as "breaking-news"—is 1%-2% of the search engine's traffic.

5. RANKING FOR RECENCY SENSITIVE QUERIES

Machine-learned ranking refers to the automatic construction of a ranking function which optimizes retrieval performance metrics [4, 5, 9, 10, 12, 21, 23, 16]. Machine-learned rankers have demonstrated significant and consistent gains over many manually tuned rankers for a variety of data sets. Optimization usually is formulated as learning a ranking function from preference data in order to minimize a loss function (e.g., the number of incorrectly ordered documents pairs in the training data). Different algorithm cast the preference learning problem from different points of view. For example, RankSVM [12] uses support vector machines; RankBoost [9] applies the idea of boosting from weak learners; GBrank [23] uses gradient boosting with decision tree; RankNet [4] uses gradient boosting with neural network.

In a typical learning-to-rank framework, some query-url pairs are selected for human judgments. Each of these query-url pairs is given a grade based on the degree of relevance (e.g. bad match, excellent match, etc). The grades will be used as target values for ranking model learning. Each query-url pair is then represented by a feature vector, including link-structure-based features, content-based features, click-based features, etc.

In this section, we describe features unique to and propose several models for recency sensitive ranking.

5.1 Recency features

Intuitively, the recency of a web page can be represented by the elapsed time, which is defined as

$$\Delta t = t_{\text{query}} - t_{\text{page}}, \tag{1}$$

where t_{query} is the time at which the user issued the query and t_{page} is the *page time*, which can be either page creation time or page content time. This elapsed time is also called *page age*. For some specific categories of web pages, the page creation time and/or content time can be easily determined. For example, for a research paper web page or a news article web page, the publication date may be extracted and used as the page time.

However, page time in general cannot be extracted reliably. For a web page, there are two types of sources that may provide evidence for page time [17]: *content-based evidence* and *link-based evidence*. Content-based evidence means that a timestamp can be extracted from page content; however, most web pages do not contain a timestamp. Link-based evidence means that the page discovery time based on the link relationship on the web can be utilized; however, page discovery time is not equal to page creation time.

To best represent page recency with available information, we propose four categories of recency-related features: *timestamp features*, *linktime features*, *webbuzz features* and *page classification features*. For timestamp features and linktime features, time values are extracted from web page content and page discovery log respectively; these two categories of features are used as an attempt to represent page freshness. Webbuzz features represent the update rates of pages and links, reflecting the recent popularity of the pages. Page classification features we use are not recency features but are closely related to page recency, and are useful in ranking model learning to help us appropriately use recency features for different classes of pages.

Each of these recency-related features may not directly and accurately represent page freshness, but they do provide useful evidence. At the ranking model learning stage, these weak, discriminant recency-related features can be combined together to better represent a page's freshness. The main ideas behind these four feature categories are introduced below; some details are omitted due to space limitations.

5.1.1 Timestamp features

Timestamp features are the time values extracted from page con-

tent. These features are especially useful for news article pages containing timestamps in some common formats. There are three steps for extracting these features:

Step 1. Detection: in a given document, we detect timestamps in common formats using regular expressions, including 1) numeric date formats including MM/DD/YYYY, DD/MM/YYYY, YYYY/MM/DD and YYYY/DD/MM formats (e.g. "09/11/2001" or "2001-9-1", but not "11/5/05" (2-digit year), "11-2005" (MM-YYYY), or "2005"); 2) English date formats such as 1st Oct 07, 1 December 2007, Jan 2nd 2008, Feb 2 06.

Step 2. Aggregation: multiple timestamps may be detected from a web page. We compute the statistics of these timestamps in an attempt to reflect page freshness from different facets, including the count of timestamps, the first timestamp (in word-view order in the page), the minimal timestamp (corresponding to the oldest), the maximal timestamp (corresponding to the newest), the mean and standard deviation of the timestamps.

Step 3. Age computation: for the timestamp statistics (except standard deviation) obtained in Step 2, we compute their corresponding page age features as their differences from query issue time.

5.1.2 Linktime features

Linktime features are extracted from a link-based page discovery log to provide page freshness evidence. Whenever a link on the web is discovered, the system records the time of this discovery with the inbound URL. The link is coded as *external* or *internal*, depending on whether or not the outbound url and inbound url are from the same domain. Similar to timestamp features, we compute the statistics of the link times for the page (url), including count, minimum, maximum, mean and standard deviation. As external links usually provide more reliable signals than internal links, we also compute the same set of discovery time statistics based on external links only. The page age features can then be computed by using query issue time.

It is possible that linktime features may not represent page freshness very accurately. For example, a link may be discovered a long time after the inbound page is created depending on crawling capability and the crawling algorithm. On the other hand, linktime features may provide strong evidence of page freshness. For example, if the discovery time of a page is three years ago, we can deduce that the page is at least three years old.

5.1.3 WebBuzz features

WebBuzz is a feature extraction algorithm that detects recently popular (including new) urls almost instantly as the crawler fetches new pages continuously. This idea is similar to page and link *activity* used in the T-Rank algorithm [3]. It works by mining data from the crawler, where each target url's inlink structure, diversity, and recency, etc. are extracted and urls are selected based on these features, and we have simple yet efficient techniques for avoiding spam and low-quality urls. The buzzing urls so identified are generally of good quality, and since the resulting features are indexed with little latency, the output features contain valuable recency information. There are three main output features: one composite buzz score which represents the intensity of web buzz, and two date features which allow for web buzz expiration: one indicates the time of buzz and the other indicates a trust-weighted inlink discovery time.

5.1.4 Page classification features

Page classification features are not time-related features, but they may be helpful for recency boosting. For example, if a page is classified as a news article page, and other recency features imply this page is very old, the ranking score of this page should be demoted. We provide these features to the learning algorithm, which is expected to learn such rules automatically. We use three classification features for each page: news page confidence, blog page confidence and page quality confidence. The confidence values are computed based on document content.

Note that there may be missing feature values for some pages. For example, a page may not contain a timestamp so that its timestamp features are invalid. In this case, we assign a default constant value to the feature. For the invalid page age feature, we set a very large positive constant value (e.g., 10,000,000 days) to it. A caveat is that the constant value zero or a small value cannot be used as default values for page age features because zero or small values mean the page is very fresh, which may mislead the learning for recency improvement.

5.2 Recency modeling

Insufficient recency ranking data is a critical problem for the learning-to-rank approach. In this section, we discuss how to appropriately utilize available data and features to solve this problem.

The most straightforward approach is the *dedicated modeling approach*, in which we train the recency ranking model using only recency ranking data, with the feature set being the combination of regular ranking features and recency features. However, recency ranking data is usually insufficient because its collection is costly and time-consuming as we have discussed in Section 3.

Another data source is regular ranking data, which is the training data used for optimizing overall relevance of result ranking. The training data sets used under the framework of learning-to-rank [16] usually belong to this category. Recency ranking data and regular ranking data are different in three aspects: 1) data: queries in recency ranking data are only recency queries while queries in regular ranking data can be any query; 2) features: regular ranking data may not have valid or accurate recency features values. For example, at the time the recency ranking data was collected, the query issue time was not accurately recorded. Therefore, the feature values of page ages cannot be calculated accurately by (1). 3) judgement: the judgement of recency ranking data is done using the recency demotion guidelines (Section 3), while regular ranking data is judged using different guidelines, which usually treat recency criteria differently. For example, some relevance judgments may take recency into account but recency demotion is neither consistently or explicitly applied as done for recency ranking data.

Despite the fact that regular ranking data is different from recency ranking data, it is still a useful information source that can help to learn a recency ranking model, because it represents overall relevance, which is also the primary objective for recency ranking model. Furthermore, the amount of regular ranking data is much larger than recency ranking data, which implies the overall relevance of the model derived from regular ranking data should be much better than that derived from recency ranking data.

As discussed above, the information provided by regular ranking data can be utilized to improve the overall relevance of a recency ranking model. Next, we seek an appropriate way to incorporate relevance information into the learning framework, so that the freshness of ranking results can be improved while preserving relevance performance. To achieve this goal, we propose three types of models: *compositional model*, *over-weighting model* and *adaptation model*, which are introduced below and will be explored more deeply based on the experimental results in Section 7.

We use the GBrank algorithm [23] for ranking model learning, as GBrank is one of the most effective learning-to-rank algorithms. For the purpose of better readability, we briefly introduce the basic

idea of GBrank. For each preference pair $< x, y >$ in the available preference set $S = \{< x_i, y_i > | x_i \succ y_i, i = 1, 2, ..., N\}$, x should be ranked higher than y. In GBrank algorithm, the problem of learning ranking functions is to compute a ranking function h , so that h matches the set of preference, i.e, $h(x_i) \geq h(y_i)$, if $x \succ y$, $i = 1, 2, ..., N$ as many as possible. The following loss function is used to measure the risk of a given ranking function h.

$$R(h) = \frac{1}{2} \sum_{i=1}^{N} (\max\{0, h(y_i) - h(x_i) + \tau\}^2) \quad (2)$$

where τ is the margin between the two documents in the pair. To minimize the loss function, $h(x)$ has to be larger than $h(y)$ with the margin τ, which can be chosen to be a constant value, or as a value varying with the document pairs. When pair-wise judgments are extracted from editors' labels with different grades, pair-wise judgments can include grade difference, which can further be used as the margin τ.

5.2.1 Compositional model

The relevance of the ranking model derived from regular ranking data is already good. One way to exploit this regular ranking model is to use its output ranking score as a feature during learning. More specifically, the training data used for recency model learning is still recency ranking data, while the features include the recency features plus the ranking score of the regular ranking model. As the output model consists of another model's output, we call it a compositional model. In this way, the recency ranking model is using the output of the regular ranking model, which captures the relevance information of query-url pairs. At the same time, the feature dimensionality is significantly reduced, so we expect the overfitting problem to be alleviated. In terms of efficiency, compositional model training is fast because the training data size has relatively small and the feature dimensionality is low.

5.2.2 Over-weighting model

The over-weighting approach combines regular ranking data and recency ranking data together as training data, and empirically determines the relative weights for these two data sources. The motivation for the over-weighting modeling approach is to leverage all available ranking data. While the grading guideline for regular ranking data is different from the recency demotion guideline for recency ranking data, we attempt to over-weight recency ranking data so that ranking freshness is emphasized while good relevance is kept.

The feature set for the over-weighting model consists of both regular ranking features and recency features. As there is sufficient training data, the ranking model learning algorithm can pick up discriminant features without much overfitting caused by high feature dimensionality.

When combining recency preference pairs and regular preference pairs, we can use different relative weights for these two data sources. The loss function becomes

$$\begin{aligned} R(h) = & \frac{w}{N_{\text{recency}}} \sum_{\langle x_i, y_i \rangle \in D_{\text{recency}}} (\max\{0, h(y_i) - h(x_i) + \tau\}^2) \\ & + \frac{1-w}{N_{\text{regular}}} \sum_{\langle x_i, y_i \rangle \in D_{\text{regular}}} (\max\{0, h(y_i) - h(x_i) + \tau\}^2) \end{aligned} \quad (3)$$

where w is used to control the relative weights between recency training data and regular training data, N_{recency} is the number of recency training pairs, and N_{regular} is the number of regular training pairs.

5.2.3 Adaptation model

Tree adaptation modeling aims to utilize the training data from one major web search domain (generic relevance ranking in our case) to help train a ranking function for a specific search domain which has insufficient training data (ranking for recency queries). Regression trees, which form the base model, are first trained using the generic relevance data. The base model is then modified according to the small amount of training data from recency ranking. Several modifications can be made on the base model: the splitting value of the feature of a node in the base model can be changed; the response value of a decision tree can be changed, and new decision trees can be appended to the base model. The assumption behind the adaptation approach is that the general ranking features can be correctly captured using a set of general ranking data, and they do not change much from a domain (general domain) to another (recency ranking). However, some details need to be adapted to the new set of data. This assumption is sound in our case, as recency queries are a special type of queries, which also bear the characteristics of general queries. Therefore, the base model can naturally be used (after adaptation) for these queries.

Adaptation also allows us to avoid the problem of insufficient training data. In practice, we have a large amount of general ranking data, but much less recency ranking data. Using only the small amount of recency data would result in a poor model. Adaptation solves this insufficient data problem by exploiting a model trained on a large set of data.

In our case, we use GBrank as the base model. We use pair-wise preference data to adapt the base model. This adaptation process tries to optimize the ranking order so as to approach the correct order as much as possible. It uses a loss function, which is defined on the wrongly ordered document pairs. The goal of the adaptation process is to minimize the loss function. This process can use any available pair-wise training data: document-query pairs judged by human editors and click preference data from the users.

Table 2 summarizes the modeling approaches that we have discussed.

6. METHODS AND MATERIALS

6.1 Data

We totally collect 70,131 query-url pairs during a period of four months (Feb.∼May, 2009) for recency data, all of which were judged by our recency demotion guidelines. For each query-url pair, its judgement and feature extraction were done within the same day. We randomly split the recency data into training set and testing set: 41,678 pairs are for training, and 28,453 pairs are for testing.

As we discussed in Section 5.2, our modeling algorithms seek help from regular ranking data, for which we collect 282,927 query-url pairs, whose recency features may not be accurate because their values were not saved timely. Also, the judging guideline does not consistently take recency into consideration. We use 206,249 pairs for training, and 76,678 for testing.

6.2 Evaluation

We use *discounted cumulative gain* (DCG) [11] to evaluate the quality of a ranking model, which is defined as

$$\text{DCG}_n = \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)} \quad (4)$$

where i is the position in the document list, G_i is the function of relevance grade. Another metric is *normalized discounted cumula-*

Table 2: Summary of recency modeling approaches. D_{recency}: recency ranking data; D_{regular}: regular ranking data; F_{recency}: recency ranking features; F_{regular}: regular ranking features (not including F_{recency})

	Data	Feature	Algorithm
dedicated model	D_{recency}	$F_{\text{recency}} + F_{\text{regular}}$	GBrank
compositional model	D_{recency}	F_{recency} + regular model score	GBrank
over-weighting model	$D_{\text{recency}} + D_{\text{regular}}$	$F_{\text{recency}} + F_{\text{regular}}$	GBrank
adaptation model	D_{recency}	F_{recency}	adaptation

Table 3: Different DCG evaluations.

$\text{DCG}_{\text{regular}}$	on regular ranking data, represents overall relevance but recency is neither consistently taken into consideration nor decoupled from non-recency relevance.
$\text{DCG}_{\text{nodemote}}$	on recency ranking data, represents non-recency relevance.
$\text{DCG}_{\text{demote}}$	on recency ranking data, represents both recency and non-recency relevance.

tive gain (NDCG), which is defined as

$$\text{NDCG}_n = Z_n \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)} \quad (5)$$

where Z_n is a normalization factor, which is used to make the NDCG of ideal list be 1. We use NDCG_1 and NDCG_5 to evaluate the ranking results. Plus, we use relative DCG_5 gain to compare two ranking results.

For different data sets, DCG values are computed based on different judging guidelines, which are listed in Table 3. $\text{DCG}_{\text{regular}}$ represents overall relevance on regular ranking data. We can compute DCG values using either the original grades (by non-recency relevance) or final grades, denoted as $\text{DCG}_{\text{nodemote}}$ and $\text{DCG}_{\text{demote}}$, respectively. Our goal is to optimize $\text{DCG}_{\text{demote}}$ for ranking results. The pure recency metric can be represented by the difference between $\text{DCG}_{\text{nodemote}}$ and $\text{DCG}_{\text{demote}}$. For a given $\text{DCG}_{\text{nodemote}}$, the lower the value of $\text{DCG}_{\text{demote}}$, the worse the freshness of the ranking result is.

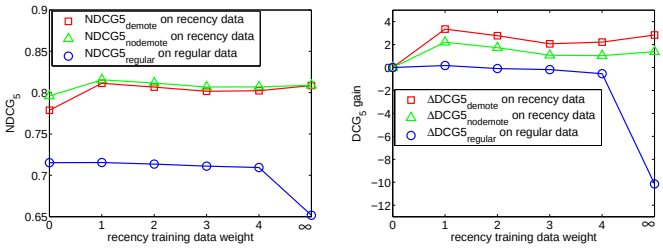
We do both offline and online experiments to test the efficacy of the modeling approaches. Offline experiment means training and validating models using ranking data, and online experiments means testing the models (derived from offline experiments) on search engine.

7. RESULTS

7.1 Offline results

Table 4 shows offline results by different recency ranking models.

First, the compositional model is better than dedicated model in sense of $\text{DCG}_{5,\text{regular}}$. This is because the relevance score is used as a feature, which significantly helps to improve relevance results on recency testing data. Second, in the sense of DCG values on regular testing data, over-weighting model gives the best result. Third, adaptation model dominates in the sense of DCG values on recency testing data, and it also provides good improvement on regular testing data. Below we analyze over-weighting model and adaptation models with more details.



(a) ndcg_5 (b) dcg_5 gain (%)

Figure 2: DCG results of over-weighting models using different recency weight values.

7.1.1 *Over-weighting model results*

Figure 2 shows the over-weighting models' results on both regular testing data and recency testing data with different weights. $\text{DCG}_{\text{regular}}$ on regular testing data mainly represent the overall relevance of the model, while $\text{DCG}_{\text{demote}}$ on recency testing data represent both recency and relevance. The gap between $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$ represents recency performance.

It is observed that if only regular training data is used to train the model (recency training data weight is 0), $\text{DCG}_{\text{regular}}$s on regular testing data are high and $\text{DCG}_{\text{demote}}$s and $\text{DCG}_{\text{demote}}$s on recency testing data are low. This is because the distribution of recency ranking data is different from that of regular ranking data (more detailed interpretation will be presented in Section 8.1). Furthermore, we observe the gap between $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$ is large, which implies the ranking recency is bad. This is expected because the judging guideline on regular ranking data does not consistently take recency into consideration.

When the recency training data is used (weight > 0), the DCG values on recency testing data (both $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$) become higher than the case that weight is 0. With the recency training data being given more weight, $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$ values remain similar and $\text{DCG}_{\text{regular}}$ values become lower. This means overall relevance becomes worse although $\text{DCG}_{\text{demote}}$ values do not change significantly. The gap between $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$ becomes smaller, which means the ranking recency is improved.

When only recency training data is used (weight is ∞, i.e., recency dedicated modeling), the DCG values on recency testing data are high. Furthermore, there is no gap between $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$, i.e., the ranking recency is good. However, the overall relevance becomes bad as $\text{DCG}_{\text{regular}}$ values are low, this is due to the overfitting on recency ranking data.

To summarize, we need to select models which give good DCG values on both recency testing data and regular testing data; at the same time, the gap between $\text{DCG}_{\text{demote}}$ and $\text{DCG}_{\text{nodemote}}$ should be small. From Figure 2, the candidate models can be those with weights being 1.0, 2.0 and 3.0.

Table 4: Offline NDCG results by different recency ranking models.

	$NDCG_{5,regular}$	$NDCG_{5,demote}$	$NDCG_{5,nodemote}$
dedicated model	0.6517 (0.0%)	0.8086 (0.0%)	0.8089 (0.0%)
compositional model	0.6984 (7.2%)	0.8047 (−0.6%)	0.8087 (−0.02%)
over-weighting model weight= 1.0	0.7154 (9.8%)	0.8113 (0.3%)	0.8155 (0.8%)
over-weighting model weight= 2.0	0.7136 (9.5%)	0.8066 (−0.2%)	0.8116 (0.3%)
over-weighting model weight= 3.0	0.7110 (9.1%)	0.8016 (−0.9%)	0.8068 (−0.3%)
adaptation model	0.6772 (3.9%)	0.8763 (8.4%)	0.8864 (9.6%)

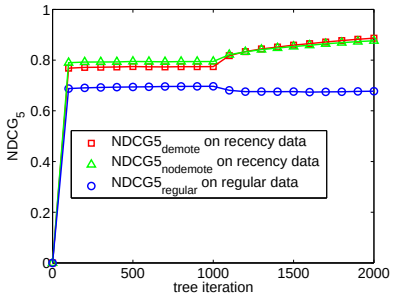


Figure 3: Testing results of adaptation model.

7.1.2 Adaptation model results

Figure 3 shows the testing results by adaptation model on both recency testing data and regular testing data. The base model consists of 1000 trees, and adaptation appends another 1000 tree in order to minimize recency loss. As we can see, there is a large jump in effectiveness for recency testing data when additive trees are appended. However, we also need to consider regular testing data, on which the NDCG values become lower. This implies that, with more additive trees, the model is more overfitted on recency ranking data because only the recency training data is used to train the additive trees; however, the overall relevance may be hurt in sense of the NDCG values on regular testing data.

7.2 Online results

We performed three days of online tests on 05/29/2009 (150 queries), 06/02/2009 (150 queries) and 06/05/2009 (150 queries). We compare different models with a baseline model, which is derived from regular ranking data. Therefore, the baseline has good relevance but bad recency. Note that this baseline model is different from the baseline model (dedicated model) used in offline experiments. Table 5 and Table 6 show the online testing results with NDCGs and DCG gains respectively. For each model category, we present the model with the best DCG results. The over-weighting model with weight being 2.0 is the best among its category. Due to system issues, we were not able to use compositional model for this online test. However, based on the offline experiments, the compositional model does not give the best results. Therefore, we mainly explore over-weighting model and adaptation model.

We observe that over-weighting model is consistently stronger

Table 5: Online NDCG results by different recency ranking models.

(a) only on recency queries

	$NDCG_{5,nodemote}$	$NDCG_{5,demote}$
baseline model	0.9111 (0.0%)	0.8941 (0.0%)
over-weighting model	0.9160 (0.5%)	0.9041 (1.1%)
adaptation model	0.9094 (−0.2%)	0.8948 (0.1%)

(b) on all queries classified as recency queries

	$NDCG_{5,nodemote}$	$NDCG_{5,demote}$
baseline model	0.9092 (0.0%)	0.8931 (0.0%)
over-weighting model	0.9143 (0.6%)	0.9025 (1.1%)
adaptation model	0.9067 (−0.3%)	0.8931 (0.0%)

Table 6: Online DCG gains over baseline by different recency ranking models.

(a) only on recency queries

	$\Delta DCG_{5,nodemote}$	$\Delta DCG_{5,demote}$
over-weighting model	1.7%	5.8%
adaptation model	−0.5%	4.9%

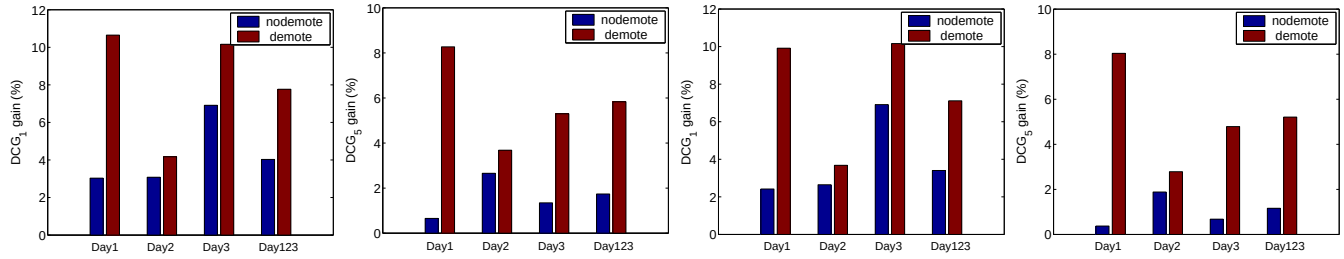
(b) on all queries classified as recency queries

	$\Delta DCG_{5,nodemote}$	$\Delta DCG_{5,demote}$
over-weighting model	1.1%	5.2%
adaptation model	−1.2%	4.2%

than baseline model in sense of both NDCG and DCG gain. By $NDCG_{nodemote}$ computed by the non-demotion grades, the NDCG gain is limited, which means the non-recency-related relevance is not improved much. However, by $NDCG_{demote}$, the NDCG gain is significantly better than baseline, which means recency ranking is significantly improved. Adaptation model is better than baseline model in sense of $\Delta DCG_{5,demote}$ (Table 6); however, its $\Delta DCG_{5,nodemote}$ is lower than that of baseline model, which suggests that the adaptation model improves recency while hurts non-recency relevance a little bit. The models' NDCG improvements are not as significant as DCG improvements, which implies that the ranking improvement mainly comes from easy search cases (the queries that can return many good results). For the hard search cases (the queries that return many bad results), it is improvement may need to rely on crawler improvement because currently there are limited good urls that have been discovered by the search engine.

Figure 4 shows the DCG gains for each of the three day's testing results by over-weighting model. Consistent trends are observed that 1) $DCG_{nodemote}$ gains are always positive, which means the overall non-recency relevance is better than baseline; 2) DCG_{demote} gains are always better than $DCG_{nodemote}$ gains, which means that ranking recency is improved 3) the non-recency queries caused by inaccuracy of the query classifier dilute the DCG gains only slightly, which means the precision of query classification is good.

Table 7 shows an example of the search results by baseline model and over-weighting model. By non-recency-demotion grades, the results by the two models are similar, which means their non-recency relevances are close. However, when recency is taken into account, the over-weighting model is better than baseline model, whose 3rd and 4th ulrs are so outdated that their grades are demoted by 2 (from excellence to fair).



(a) DCG_1 gains only on recency queries (b) DCG_5 gains only on recency queries (c) DCG_1 gains on all queries (d) DCG_5 gains on all queries

Figure 4: Online DCG gains by over-weighting model over baseline model on different test days.

Table 7: An example of ranking recency improvement. The query is "mars rover name".

(a) ranking result by baseline model

	url	$grade_{nodemote}$	$grade_{demote}$
1	http://marsrovername.jpl.nasa.gov/	excellent	excellent
2	http://marsprogram.jpl.nasa.gov/	good	good
3	http://www.lego.com/rovers/default.asp	excellent	fair
4	http://mars.jpl.nasa.gov/MPF/rover/name.html	excellent	fair
5	http://www.mars-rover.com/	bad	bad

(b) ranking result by over-weighting model

	url	$grade_{nodemote}$	$grade_{demote}$
1	http://marsrovername.jpl.nasa.gov/	excellent	excellent
2	http://mpfwww.jpl.nasa.gov/	good	good
3	http://www.nasa.gov/home/hqnews/2009/may/HQ_09-122_MSL_named_Curiosity.html	good	good
4	http:/www.jpl.nasa.gov/news/news.cfm?release=2009-089	excellent	excellent
5	http://www.jpl.nasa.gov/missions/mer/	excellent	excellent

Table 8: Indirect comparison of regular ranking data distribution and recency ranking data distribution. The table slot values are $NDCG_{5,\textbf{nodemote}}$s, which are obtained by the training and testing sets specified in the table.

	recency testing data	regular testing data
recency training data	0.7857	0.6497
random training data	0.7569	0.6865

8. DISCUSSIONS

Previous experiments have shown the efficacy of our learn-to-rank approach for recency improvement. In this section, we further analyze the critical factors in this approach, so that we can better understand the results and improve them in future.

8.1 Data distribution

As we have observed in offline and online results, using recency training data may help to improve $NDCG_{5,nodemote}$ on recency testing data. This leads to the question: in feature space, if we do not consider recency factor, what is the distribution difference between recency ranking data and regular ranking data?

Recency query set is a subset of all queries. Intuitively, many categories of queries can potentially be recency queries. For example, a celebrity name query can be a recency query during a short period of time when there are news related to this celebrity. Similarly, product queries, news-event queries and many other queries can be recency queries. On the other hand, there are some categories of queries that are unlikely to be recency queries, e.g., domain queries.

Table 9: The average ranks of the top 5 recency features, and their average relative feature importance scores in over-weighting models with different recency training data weight. Recency training data weight is denoted by wt.

wt	avg. rank	avg. score
0.0	63.1	12.0
1.0	57.2	13.0
2.0	36.6	16.7
3.0	23.9	22.8
4.0	19.5	26.8

Due to the high dimensionality of ranking features, it is difficult to directly compute the distribution difference between these two types of data. Instead, we use different training/testing set combinations to explore the distribution difference. As introduced in Section 6.1, there are 41,678 query-url pairs in recency training data. We randomly select the same amount of query-url pairs from regular training data, and we call the selected data as *random training data*. We train two ranking models using recency training data and random training data respectively, and apply them to recency testing data and regular testing data to compare NDCG values. During model training, recency features are excluded because our purpose is to explore the non-recency-related distribution. Table 8 shows that 1) on recency testing data (i.e., for recency ranking problem), the model derived from recency training data is better than that derived from random training data; 2) on regular training data (i.e., for generic ranking problem), the model derived from random training data is better. These observations imply that there is a significant distribution difference between recency ranking data and regular ranking data. This explains why when recency training data is used, non-recency relevance on recency testing data may be improved.

8.2 Recency features

Recency features play key roles in recency model learning. We compute feature importance by the method proposed in [10], and rank the features by the descending order of the feature importance scores. The importance score of the most important feature in the whole feature set is 100.0. Table 9 shows the average ranks of the top 5 recency features, and their average relative feature importance scores in over-weighting models with different recency training data weight. The more weight of recency training data is given, the higher the average rank and the average feature importance for recency features.

Below the most important five recency features with their importance ranks in the whole feature set (recency training data weight value is 2.0):

12nd: the latest time that an external link to the url is discovered on the Web;

19th: the earliest time that the url is discovered on the Web;

22nd: the mean time that the external links to the url are discovered on the Web;

28th: the timestamp that first appears in the page content in word-view order;

29th: the earliest timestamp that appears in the page content in time order.

Therefore, linktime features are the most important recency features among all recency features.

Recency features correlate with regular ranking features. For example, compared with an old page, a fresh page usually have fewer clicks and external links. While ranking model usually should favor the pages with more links and clicks, it should also promote fresh pages for recency ranking problem. Thus, recency is competing with popularity, which is usually indicated by link-based features and click-based features. We have examined the urls pairs whose orders contradict with ground-truth labeling, and we find that the click-based features are over-aggressive for most of the cases. This leads to the interesting topic on how to appropriately deal with the relationship between recency and popularity, and we will study modeling approaches which can better exploit recency features and popularity-related features.

9. CONCLUSIONS

We construct a web search system that improves ranking recency while preserves good overall relevance. We propose a query classification algorithm that can automatically detect recency queries with high precision; the detected recency queries are applied with the recency ranking model. With recency demotion judgement guideline, we incorporate recency into the optimization target. Different categories of recency features provide recency evidence and are effectively employed in ranking model learning. To solve the recency data insufficiency problem, we explored several modeling approaches by utilizing regular ranking data, which helps to achieve good ranking recency as well as good relevance.

While the work in this paper proves that ranking recency can be improved under learning-to-rank framework, there is much more room for further improvement along this direction. For example, from click log, we can either extract click-based recency features to represent the urls' recent popularity, or extract click preference data to enhance recency training set. We will study these utilities of click log to improve recency ranking. The current recency demotion judgement guideline is based on heuristic studies so that the demotion may be biased for some cases. We need to explore this issue in a systematic way so that the demotion may be more appropriate for users.

10. REFERENCES

[1] I. Arikan, S. Bedathur, and K. Berberich. Time will tell: Leveraging temporal expressions in ir. *WSDM*, 2009.

[2] R. Baeza-Yates, F. Saint-Jean, and C. Castillo. Web dynamics, age and page qualit. *String Processing and Information Retrieval*, pages 453–461, 2002.

[3] K. Berberich, M. Vazirgiannis, and G. Weikum. Time-aware authority rankings. *Internet Math*, 2(3):301–332, 2005.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Proc. of Intl. Conf. on Machine Learning*, 2005.

[5] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise. *Proceedings of ICML conference*, 2007.

[6] J. Cho, S. Roy, and R. Adams. Page quality: In search of an unbiased web ranking. *Proc. of ACM SIGMOD Conference*, 2005.

[7] G. M. Del Corso, A. Gulli, and F. Romani. Ranking a stream of news. *Proc. of WWW Conference*, 2005.

[8] F. Diaz. Integration of news content into web results. *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM)*, pages 182–191, 2009.

[9] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Proceedings of International Conference on Machine Learning*, 1998.

[10] J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Statist.*, 29:1189–1232, 2001.

[11] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20:422–446, 2002.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[13] J. Kleinberg. Bursty and hierarchical structure in streams. In *KDD*, pages 91–101, 2002.

[14] A. C. König, M. Gamon, and Q. Wu. Click-through prediction for news queries. *Proc. of SIGIR*, pages 347–354, 2009.

[15] X. Li, B. Liu, and P. Yu. Time sensitive ranking with application to publication search. *Proceedings of Eighth IEEE International Conference on Data Mining*, pages 893–898, 2008.

[16] T. Y. Liu. Learning to rank for information retrieval. *Tutorial on WWW conference*, 2009.

[17] S. Nunes. Exploring temporal evidence in web information retrieval. *BCS IRSG Symposium: Future Directions in Information Access*, 2007.

[18] S. Pandey, S. Roy, C. Olston, J. Cho, and S. Chakrabarti. Shuffling a stacked deck: The case for partially randomized ranking of search engine results. *VLDB*, 2005.

[19] M. Pasca. Towards temporal web search. *ACM SAC*, 2008.

[20] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, pages 131–142, 2004.

[21] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *Proceedings of the 30th ACM SIGIR*, 2007.

[22] R. Zhang, Y. Chang, Z. Zheng, D. Metzler, and J. Nie. Search result re-ranking by feedback control adjustment for time-sensitive query. *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2009.

[23] Z. Zheng, H. Zhang, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. *NIPS*, 2007.