Learning to Rank for Freshness and Relevance

Na Dai
Computer Sci. & Engr.
Lehigh University, PA, USA
nad207@cse.lehigh.edu

Milad Shokouhi
Microsoft Research
Cambridge, UK
milads@microsoft.com

Brian D. Davison
Computer Sci. & Engr.
Lehigh University, PA, USA
davison@cse.lehigh.edu

ABSTRACT

Freshness of results is important in modern web search. Failing to recognize the temporal aspect of a query can negatively affect the user experience, and make the search engine appear stale. While freshness and relevance can be closely related for some topics (e.g., news queries), they are more independent in others (e.g., time insensitive queries). Therefore, optimizing one criterion does not necessarily improve the other, and can even do harm in some cases.

We propose a machine-learning framework for simultaneously optimizing freshness and relevance, in which the trade-off is automatically adaptive to query temporal characteristics. We start by illustrating different temporal characteristics of queries, and the features that can be used for capturing these properties. We then introduce our supervised framework that leverages the temporal profile of queries (inferred from pseudo-feedback documents) along with the other ranking features to improve both freshness and relevance of search results. Our experiments on a large archival web corpus demonstrate the efficacy of our techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

Temporal profiles, Query classification, Freshness ranking

1. INTRODUCTION

The query stream seen by a web search engine and the interpretation of those queries change over time. Previous analysis has shown that web logs clearly reflect daily events in user queries [6]. For example, during seasonal events such as Halloween, there are always spikes in the frequency of related queries such as "halloween", "halloween costumes" and "pumpkins". For many of the queries that correspond to events, the best answer may change over time (e.g., the latest SIGIR conference homepage for the query "sigir conference"). In more extreme cases, the major intent behind the same query can temporally vary; for instance, the query "US open" is more likely to be targeting the tennis open in September, and the golf tournament in June. Kulkarni et al. [22] referred to this class of temporally ambiguous queries as *shift* topics.

News events, depending on their significance, can cause enormous growth in frequency of related queries.[1] It is also not uncommon for news events to change the general meaning of a query. For example, the query "ipad" which could be treated as a misspelling for "ipod" in 2009, suddenly turned into a valid query with several related websites in 2010.[2] Therefore, making search engine results appear current and fresh is important to satisfy users' ever-changing information needs.

In this paper, we focus on improving the ranking of results for queries based on their temporal profiles. Of course, the importance of the temporal profiles of queries extends beyond web result ranking; advertisement rankers have to address similar problems; related search and auto-complete suggestions must provide users with fresh and relevant alternatives to their queries; vertical search [11] ranking and triggering can be affected by temporal changes, and in general, the entire search experience can be influenced according to the temporal aspect of a query.

Learning ranking functions that can respond effectively to diverse temporal dynamics of queries is challenging. One of the difficulties is that traditional machine learning ranking algorithms fail to consider the interaction between freshness and relevance. While relevance clearly quantifies the topical matchability between query and web pages, freshness can be interpreted in different ways. For certain *temporal* queries such as breaking news, freshness is more meaningful when the actual page content reflects new information. Whereas, for *non-temporal* (time-insensitive) queries, it makes more sense to interpret freshness as the recency of page maintenance with respect to the time point of generating ranking lists (suppose web pages contain such information). Therefore, these two interpretations for freshness may be correlated to some extent but are not the same, considering that pages updated recently tend to record fresh information. It is worthwhile pointing out that both explanations can be part of the overall quality of search results that influences user search experience. In this work, the definition of freshness is sensitive to query temporal characteristics, varying

SIGIR'11, July 24–28, 2011, Beijing, China.

[1] For example, the traffic caused by queries related to Michael Jackson's death in 2009, was so huge that Google mistook it as an attack (Source: Google Blog, 26 Jun 2009).

[2] It is probably still the case that some people mistype ipod as ipad. However, this group no longer represents the majority.

on whether human editors (judges) can identify temporal intents concealed within queries. (See Section 4 for details.)

For certain *temporal* queries such as breaking news, relevance and freshness are highly correlated. Therefore, a ranker optimized for returning fresh documents may produce satisfactory results. However, for queries that are not usually time-sensitive (e.g., "facebook", "machine learning"), paying too much attention to freshness may significantly hurt ranking effectiveness in terms of relevance. Among common ranking features, clicks, anchor-text and historical data might be the most powerful for answering time-insensitive queries. For temporal queries however, other features such as the rate of content change in documents may provide better signals [22]. Therefore, a ranker optimizing either freshness or relevance only may not be flexible enough to deal with the temporal dynamics of queries effectively.

To address this issue, previous work [3, 12] suggested training separate rankers for different classes of queries. The query is first classified according to its temporal profile, and then is sent to the appropriate ranker that has been optimized for either relevance or freshness. The main disadvantage of classification-based techniques is that selecting a wrong ranker due to misclassification can significantly degrade the performance.

We propose a machine learning model that optimizes freshness and relevance simultaneously. Our flexible framework allows training multiple rankers with different optimization functions, and runs each query against all rankers with the weights varying according to the query's temporal profile. This is in contrast with existing solutions that suggest selecting one ranker per query, and consequently has a lower risk of poor performance when queries are misclassified. In addition, instead of splitting the labeled data to train separate rankers, our technique leverages the entire data in training all rankers. To the best of our knowledge, this is the first attempt to incorporate the trade-off between freshness and relevance into a single ranking framework.

Our work can be regarded as an extension to the family of *divide and conquer* (DAC) techniques for ranking [2]. In DAC, queries are clustered based on their feature representations, and separate rankers are trained with each for one cluster simultaneously. At test time, the query is compared against the generated cluster centroids and is ranked under all rankers with the weights depending on query-cluster similarity values. We follow a similar path since DAC enables specialized ranker training by considering query features, but we incorporate multiple criteria (freshness and relevance) into ranking optimization. We also modify the DAC loss function by introducing a new query-document importance factor that emphasizes certain documents during training, and leads to further improvements in the results. Our experiments on a large web archive demonstrate that the rankers trained by our techniques can achieve better relevance and freshness compared to state-of-the-art alternatives. The contributions of this paper are four-fold:

1. We extend an existing learning to rank framework to optimize for both freshness and relevance.

2. We introduce a new loss function that emphasizes certain query-document pairs for better optimization.

3. We investigate the correlation between freshness and relevance and compare it across temporal and non-temporal queries.

4. We introduce *hybrid NDCG*, a new variant of NDCG that considers both freshness and relevance labels in evaluation.

The remainder of this paper is organized as follows. We review prior work in Section 2, and continue by introducing our criteria-sensitive ranking specialization framework in Section 3. Features are summarized in Section 4. We describe our experimental results in Section 5 followed by a more thorough discussion in Section 6. Section 7 concludes the paper and suggests directions for future research.

2. RELATED WORK

Pairwise learning to rank techniques have been widely studied in recent years [5, 15, 19]. They cast learning to rank as a preferential relation learning problem. Given a query and a pair of associated documents, if one is more relevant than the other, then it is boosted in the training process to get a higher rank. In most early work, the query type information was ignored in ranking, which limits the effectiveness of ranking functions. For instance, navigational queries target specific websites, while informational queries have a broader range of relevant answers. Hence, their ranking models could be optimized in different ways that depend on query intent [21].

Query-dependent loss/ranking functions were introduced to address these issues [2, 3, 16]. The general idea is to adopt a query-dependent loss or ranking function based on the query type (class). Geng et al. [16] proposed a k-Nearest Neighbor based method which trains a query-dependent ranking function for each query based on its nearest neighbors in the training set.

Bian et al. [3] achieved better results by learning both multiple ranking functions (by minimizing query-dependent ranking risks) and query categorization (navigational, informational, transactional) simultaneously. Although the query-dependent loss function has been found superior to the query-dependent ranking method of Geng et al. [16], it still leaves a few issues unaddressed: (1) query categorization and taxonomies may not be available or could be too noisy; (2) external taxonomies may not necessarily provide the best way of splitting queries for training specialized rankers; and (3) such categories may not be fine-grained enough for training and ranking purposes. To overcome these problems, Bian et al. [2] proposed a *divide-and-conquer* framework (DAC) for ranking specialization and instantiated it on RankSVM [19].

Divide and conquer for ranking. The DAC ranking framework [2] can be summarized in three main steps: (1) identifying ranking-sensitive query categories, (2) learning topic-specific ranking models via minimizing a global ranking risk, and (3) running each unseen query against all ranking models and merging the outputs to produce the final ranked list.

In the first step (divide), queries are categorized (clustered) in a soft way to form ranking-sensitive topics. The queries in each cluster have similar ranking characteristics and similar ranking feature discriminativity. Bian et al. [2] suggested using the ranking features aggregated from the top-ranked pseudo-feedback documents that are generated by a reference ranking model to represent queries for clustering. In this way, the queries that have similar features are clustered together. The number of query clusters (topics) can be pre-defined or could be determined according to gap statistics [27].

In the second step (*conquer*), a unified learning method is used to train multiple ranking models, one for each cluster (topic). The authors applied a global loss aggregated across all ranking topics, and trained multiple rankers simultaneously by minimizing the global ranking risk. Each query contributes to training all ranking models though with the different weights that are determined by the probabilities of belonging to each cluster.

In the last step, each unseen query is submitted to all ranking

models and a weighted combination is used to merge the final results.

While most previous work focused on optimizing relevance, we propose an extended framework which optimizes freshness and relevance simultaneously in a more adaptive way. We enhance query representations by adding *criteria-sensitive* features that can capture different aspects (relevance, freshness) of query-document pairs. Each query is categorized according to both temporal and relevance features, and the final ranking is produced by merging the results generated from several different ranking models.

Multiple criteria ranking. Training ranking models for multiple criteria beyond relevance, such as diversity, freshness, and efficiency, has been the subject of many recent papers [12, 13, 17, 28]. Dong et al.'s work on recency ranking [12, 13] is among the closest to our work; they consider freshness in instance labeling for training effective ranking models. They argued that freshness is especially important for breaking news queries and demoted the relevance labels of stale pages for training. Empirical experiments demonstrated that such demotion can result in significant improvements on both relevance and freshness. We similarly generate hybrid labels for documents based on their relevance and freshness grades, and show that the labels generated by our strategy are more effective than those demoted for training. Despite this resemblance, our optimization tasks are fundamentally different; Dong et al. [12, 13] studied learning single *adaptive* or *over-weighting* rankers that can be trained with an imbalanced amount of training data for freshness and relevance primarily from the perspective of ranking adaptation. We investigate multi-criteria ranking problem in a *divide and conquer* framework with balanced distribution of training data, and emphasize adaptive balance between different criteria.

Temporal signals for ranking. Exploiting temporal signals that capture the dynamics of queries, web pages, hyperlinks, and user interaction to improve search quality has been widely studied. Several methods focused on complementing content-based matching by utilizing the knowledge of query temporal characteristics [1, 10, 20, 23, 24]. Typically this includes: (1) profiling query temporal characteristics, e.g., generating a temporal distribution over pseudo-feedback documents or based on query popularity over time; and (2) emphasizing documents whose temporal characteristics are close to the query's temporal profile, e.g., enhancing document representation by adding temporal dimension and then incorporating temporal based matching into search process.

Elsas and Dumais [14] incorporated the dynamics of content changes into document language models and showed that their enhanced representations can improve retrieval effectiveness. Dai and Davison [8] exploited the frequency of web content and hyperlink changes over time for better estimation of web authorities. Dong et al. [13] used Twitter data to detect and rank fresh documents.

3. CRITERIA-SENSITIVE RANKING

In this section, we introduce our *criteria-sensitive* divide-and-conquer ranking framework (denoted as CS-DAC) that incorporates the balance between relevance and freshness into training customized rankers that optimize both freshness and relevance.

CS-DAC framework. A typical ranking function f with ω parameters takes a query-document feature vector $\mathbf{X}$ as input and produces ranking scores of documents.

$$\hat{\mathbf{y}} = f(\mathbf{X}, \omega)$$

The common goal of learning to rank systems is to find a ranking model f^* that takes query-document feature vectors as input, and produces a document ranking—as close as possible to the *oracle* ranking of documents according to their relevance labels $\mathbf{y}$—by minimizing the ranking risk aggregated from the loss $\mathcal{L}$ of all training queries.

$$f^* = \arg\min_f \sum_q \mathcal{L}(f(\mathbf{X}_q, \omega), \mathbf{y}_q) = \arg\min_f \sum_q \mathcal{L}(\hat{\mathbf{y}}_q, \mathbf{y}_q)$$

By considering query differences in the DAC framework, we essentially *cluster*[3] training queries based on their ranking characteristics, and train one ranker per cluster. Each query contributes to learning all rankers with different importance based on its topical affinity to query clusters. Each ranker f_i^* is learned via:

$$f_i^* = \arg\min_{f_i} \sum_{q \in \mathcal{Q}} \mathcal{I}(q, i) \mathcal{L}_i(\hat{\mathbf{y}}_q, \mathbf{y}_q) \tag{1}$$

where $\mathcal{Q}$ is the training query set, and $\mathcal{I}(q, i)$ is the importance of query q with respect to the i^{th} ranking model.

To account for relevance and freshness simultaneously, we propose to use hybrid labels that are generated based on freshness and relevance judgments.[4] For this purpose, we exploit a weighted harmonic mean function which maps relevance and freshness grades (i.e., $y_{q,d}^R$ and $y_{q,d}^F$ on the query-document pair $<q, d>$) to a single equivalent numerical score $\widetilde{y}_{q,d}$ for training f_i^*. We believe harmonic mean is appropriate here since (1) it heavily biases towards the minimum score; (2) it is more sensitive when $y_{q,d}^R$ and $y_{q,d}^F$ are close; and (3) it has been shown as a good optimization metric for tasks such as learning to rank for efficiency [28] and classification. Formally, $\widetilde{y}_{q,d,i}$ is defined as:

$$\widetilde{y}_{q,d,i} = \frac{(1 + \beta_i^2) \cdot y_{q,d}^R \cdot y_{q,d}^F}{y_{q,d}^R + \beta_i^2 \cdot y_{q,d}^F} \tag{2}$$

where parameter β_i sets the trade-off between relevance and freshness for each ranker, and is learned during training. Allowing different values of β for rankers enables a flexible framework where each ranker can assign different weights to freshness and relevance. It also means that each query-document pair may affect the pairwise learning of each ranker differently.[5] Therefore, we factorize query-document pair importance as follows:

$$f_i^* = \arg\min_{f_i} \sum_{q \in \mathcal{Q}} \mathcal{I}(q, i) \times \sum_{<d_1, d_2> \in \mathcal{D}_q} \mathcal{U}'(q, i, d_1, d_2) \mathcal{L}_i\Big(\begin{bmatrix} \hat{y}_{q,d_1,i} \\ \hat{y}_{q,d_2,i} \end{bmatrix}, \begin{bmatrix} \widetilde{y}_{q,d_1,i} \\ \widetilde{y}_{q,d_2,i} \end{bmatrix}\Big) \tag{3}$$

where, $\mathcal{D}_q$ is the set of preferential query-document pairs with respect to query q, and $\mathcal{U}'(q, i, d_1, d_2)$ is the importance of $<d_1, d_2>$ in training for query q with respect to the i^{th} ranking model. For simplicity, we assume $<q, d_1>$ and $<q, d_2>$ are independent, and so factorize the importance of the preferential pair $\mathcal{U}'(q, i, d_1, d_2)$ as follows.

$$\mathcal{U}'(q, i, d_1, d_2) = \mathcal{U}(q, i, d_1) \cdot \mathcal{U}(q, i, d_2)$$

where $\mathcal{U}(q, i, d_1)$ is the importance of query-document pair

[3] We use query cluster, topic and category interchangeably.

[4] Generating hybrid labels (single aggregate objective functions), is a simple form of multi-criteria optimization [26].

[5] Similar ideas can be applied to list-wise and point-wise ranking learning algorithms.

$<q, d_1>$ in training for query q with respect to the i^{th} ranking model.[6]

Ensemble ranking. Given an unseen query q', we first profile its query characteristics, and then calculate its distances to the centroids of existing query clusters $\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_n$. The trained ranking functions are then scored according to the normalized distance between the query and their corresponding clusters (a.k.a. query importance $\mathcal{I}$), given by:

$$W_i = \frac{\mathcal{I}(q', i)}{\sum_{i'=1}^{n} \mathcal{I}(q', i')}$$

The query q' is run against all n rankers (one for each cluster), and the final results $\theta_{q'}$ are produced according to the ensemble ranking of their outputs. That is,

$$\theta_{q'} = \sum_{i=1}^{n} W_i f_i^*(\mathbf{X}_{q'}, \omega_i)$$

where f_i^* is the i^{th} ranking model, $\mathbf{X}_{q'}$ is the query-document feature vectors for query q', and ω_i is the feature weights.

The CS-DAC framework summarized in Equation 3 consists of three main factors: query importance ($\mathcal{I}$), ranker-specific query-document importance ($\mathcal{U}$), and the loss function ($\mathcal{L}$). We continue by describing each of these items.

Query importance ($\mathcal{I}$). In the *divide* step of the DAC framework, the query space is split into a few clusters based on *criteria-sensitive* features. These are the features that are extracted from the top-ranked documents of a basic reference ranker (BM25 [25] in our work) for the query. We will provide more details about these features in Section 4.

The $\mathcal{I}(q, i)$ values provide a Binomial distribution over each of criteria-sensitive query clusters, and specify the importance of different ranking functions. We use Gaussian Mixture model as soft k-means clustering to group queries into clusters. The importance of query q with respect to the i^{th} cluster is thus given by:

$$\mathcal{I}(q, i) = 1 - \frac{\|\mathbf{p}_q - \mathbf{c}_i\|^2}{\max_{q' \in \mathcal{Q}} \|\mathbf{p}_{q'} - \mathbf{c}_i\|^2} \quad (4)$$

where $\mathbf{p}_q$ and $\mathbf{c}_i$ respectively denote the feature vectors of query q and the centroid of the i^{th} cluster, and $\mathcal{Q}$ represents the set of training queries. Therefore, $\mathcal{I}(q, i)$ is scaled between $[0, 1]$, and is inversely proportional to the distance between query feature vector $\mathbf{p}_q$ and cluster centroid $\mathbf{c}_i$.

Document importance ($\mathcal{U}$). In pairwise learning to rank methods, the importance of a document with label $\mathbf{y}$ during training depends on the number of times it is compared to other documents with different labels. Due to the ranker-specific value of β which is set during training, a query-document pair with the same relevance and freshness grades can get unequal hybrid labels under different rankers, and hence may contribute unequally in training various rankers. Besides, centralizing hybrid label distribution within each query cluster stabilizes the correlation between freshness and relevance, which further emphasizes the effect of β_i in Equation 2. To factorize these impacts, we introduced the $\mathcal{U}$ component in Equation 3. We estimate the importance of a query-document pair with label $\mathbf{y}_{q,d}$ by the likelihood of visiting that label in the training dataset, under the assumption that the importance of a hybrid label is proportional to the ratio of query-document pairs with that label in the training dataset. We define the document importance $\mathcal{U}$ as below.

$$\mathcal{U}(q, i, d) = \frac{\sum_{q' \in \mathcal{Q}} N(q', i, \mathbf{y}_{q,d}) \cdot N(q', i, \neg \mathbf{y}_{q,d})}{\sum_{\mathbf{y'} \in \mathcal{Y}_i} \sum_{q' \in \mathcal{Q}} N(q', i, \mathbf{y'}) \cdot N(q, i, \neg \mathbf{y'})} \quad (5)$$

where $\mathcal{Y}_i$ is the space of labels for ranker i, and $\mathcal{Q}$ denotes the training query set. The number of documents with and without label $\mathbf{y}$ are represented by $N(q, i, \mathbf{y})$ and $N(q, i, \neg \mathbf{y})$. Equation 5 can be regarded as a function of the unique hybrid label $\mathbf{y}_{q,d}$, and is denoted as $w(\mathbf{y}_{q,d})$ for short.

There are two potential problems with this type of normalization: (1) additional inter-label dependencies may arise from comparing common labels (e.g., $\mathbf{y}_a$ and $\mathbf{y}_b$, versus $\mathbf{y}_b$ and $\mathbf{y}_c$), and, (2) overemphasizing certain documents inevitably introduces bias in ranking. To overcome these issues, we exploit a random walk approach to determine $\mathcal{U}$ (instead of Equation 5) that has the effect of smoothing document importance values.

To perform a random walk, we first construct a fully connected bipartite graph $G(V, E)$ (one graph per ranker) in which each node (state) v stands for a unique *hybrid* label $\mathbf{y}$ (associated with the weight $w(\mathbf{y})$), and each edge e is associated with a weight computed according to the number of times the labels of the connected nodes compare with each other during training. At each step, the random walk surfer jumps to a random node with probability d (selection among random nodes is proportional to $w(\mathbf{y})$ values) or follows some connected edge with probability $1 - d$ (the selection among connected edges is proportional to the weights on edges). The value of d can be pre-defined or set during the training and validation. When d equals 1, the probability that the random surfer reaches every node (state) is proportional to the direct comparison between preferential query-document pairs with different hybrid labels. Whereas, $d = 0$ suggests document importance entirely propagates through indirect comparison between preferential query-document pairs. Parameter d actually controls the extent that such propagation (from indirect comparison) influences the computation of document importance. We analyze the importance of $\mathcal{U}$, with and without smoothed probabilities in Section 6.

Loss function ($\mathcal{L}$). The core of each ranker in our CS-DAC framework is a loss function that is trained for hybrid labels (Equation 2). We follow Bian et al. [3] and use RankSVM [19] as our basic learning algorithm although it is important to note that the framework is flexible and not restricted to any particular learning technique.

RankSVM [19] is designed to maximize the margin between positively and negatively labeled documents in the training data by minimizing the number of discordant pairs. The RankSVM optimization problem is defined as:

$$\arg \min_{\omega, \xi_{q,i,j}} \frac{1}{2}\|\omega\|^2 + C \sum_{q,i,j} \xi_{q,i,j} \quad \textit{subject to}$$

$$\begin{aligned} \forall y_i^q \succeq y_j^q : \quad & \omega^T X_i^q \geq \omega^T X_j^q + 1 - \xi_{q,i,j}, \\ \forall_q \forall_i \forall_j : \quad & \xi_{q,i,j} \geq 0 \end{aligned}$$

where the non-negative slack variable $\xi_{q,i,j}$ is used to approximate the NP-hard optimization solution by minimizing the upper bound $\sum \xi_{q,i,j}$. Parameter C sets the trade-off between the training error and the margin size. The query-document feature vectors for documents i and j are respectively represented by X_i^q and X_j^q. The notation $y_i^q \succeq y_j^q$ implies that the document i is ranked higher than

[6]The independence assumption is unrealistic, but we believe it is not unreasonable because if two query-documents pairs are important, then so is their preferential pair.

document j with respect to query q in the training dataset (i has the same or higher relevance than j).

CS-DAC modified the RankSVM loss function by incorporating query importance ($\mathcal{I}$) and document importance ($\mathcal{U}$). Formally, the i^{th} ranking model of CS-DAC is optimized via:

$$\arg \min_{\omega_i, \xi_{q,j,k}} \frac{1}{2}\|\omega_i\|^2 + C \sum_{q,j,k} \xi_{q,j,k} \qquad (6)$$

$$\textit{subject to,} \quad \forall \widetilde{y}_{q,j,i} \succeq \widetilde{y}_{q,k,i} : \mathcal{I}(q,i)\mathcal{U}(q,i,j)\omega_i^T X_j^q \geq \mathcal{I}(q,i)\mathcal{U}(q,i,k)\omega_i^T X_k^q + 1 - \xi_{q,j,k},$$

$$\forall_q \forall_i \forall_j : \quad \xi_{q,i,j} \geq 0$$

where $\xi_{q,j,k}$ is the slack variable and parameter C sets the trade-off between training error and the margin size.

In CS-DAC, several rankers are trained simultaneously, and each ranking function f_k^* (see Equation 3) is optimized using the CS-DAC loss function and hybrid labels. The β values are tuned via hill climbing based on the *hybrid NDCG* values of the final ranking lists merged from different rankers. That is, each ranker is trained on different values of β and the best combination of rankers is chosen by hill climbing on the training and validation data. Here, *hybrid NDCG* extends the commonly used evaluation metric NDCG [18] to take hybrid labels for evaluation, since this new freshness-sensitive metric can take into account both freshness and relevance into a single measurement, aiming to quantify the overall search quality. Formally, we define *hybrid NDCG* as below:

$$\textit{hybrid NDCG}(n) = Z_n \sum_{j=1}^{n} \frac{2^{(\gamma \mathbf{y}_R + (1-\gamma)\mathbf{y}_F)} - 1}{\log_2(j+1)} \qquad (7)$$

where Z_n is the oracle *discounted cumulative gain* at ranking cut-off n, that bounds the NDCG values between 0 and 1. The $\mathbf{y}_R$, and $\mathbf{y}_F$ values—also known as *gains*—are assigned according to the relevance and freshness labels of documents. Parameter γ specifies the trade-off between relevance and freshness and is set to 0.5 in our experiments. Note that $\gamma = 1$ turns hybrid NDCG into typical relevance-based NDCG, while setting γ to zero, makes it the same as the NDCF metric [13]. Dai and Davison [8] also adopted NDCG with freshness labels, although they did not refer to it as NDCF. While other combination forms may better fit the search utility that quantifies comprehensive users' satisfaction, we leave the best definition of *hybrid NDCG* for future work.

4. EXPERIMENTAL SETUP

Testbed data. Standard learning to rank datasets only contain relevance judgments for query-document pairs without any information regarding their freshness.

Therefore, we built a new testbed based on a large archival web corpus. Our dataset contains 158 million unique URLs and 12 billion links from the `.ie` domain, covering the time span from January 2000 to December 2007 (one snapshot per month and 88 in total). We removed pages with less than five snapshots, and only kept the remaining 3.8 million unique pages with 435 million links in total.

We choose *April 2007* as our time point of interest for ranking evaluation. We constructed two *temporal* and *non-temporal* query sets, each containing 90 queries. While the query size is small, the queries in the temporal set are manually selected from Google Trends suggestions for Ireland, which were popular during *April 2007*.[7] For the non-temporal set, we first randomly sampled queries

[7] `www.google.com/trends`

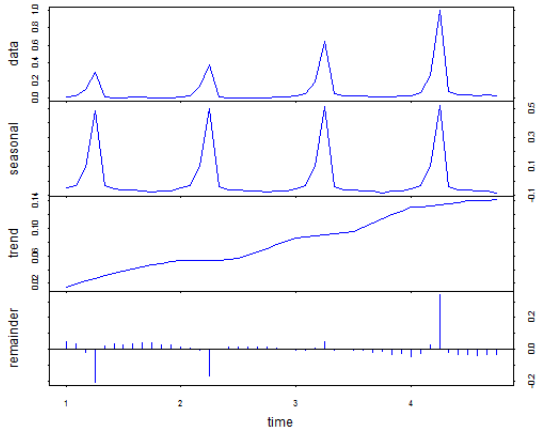


Figure 1: The STL decomposition [7] of a time series into seasonal, trend and remainder components. The data is generated from the click histogram of the query *jingle bells* in a commercial search engine.

from a 2006 MSN query log (i.e., generating a representative query sample from a real-world search log), and then automatically filtered out about 10% of them that were detected as *potentially temporal* by a commercial classifier. The classifier has high precision (almost all Google Trend queries are detected as temporal), and uses several years of the query-frequency history extracted from the query logs of a major commercial search engine.

Judgments and metrics. We have an average of 71 URLs per query judged by one or more participants from Amazon Mechanical Turk.[8] Given a query-URL pair, the judges were instructed to assess the quality of the URL with respect to both relevance and freshness. For relevance, the selection was among *highly relevant*, *relevant*, *borderline*, *not relevant* and *not related*, which was further translated to integer gains ranging from 4 to 0. For freshness, editors were instructed to judge the URL freshness for the given query according to our chosen point in time (April 2007).[9] Judges could select between *very fresh*, *fresh*, *borderline*, *stale*, and *very stale*, which we transferred into $\{4, 3, 2, 1, 0\}$. Judges were also required to provide the confidence of their judgements by choosing between *high*, *medium* and *low*. Table 2 shows the guideline of query-URL pair judgments used by Mturk workers. Judgments with low confidence were resubmitted for labeling. The standard deviations of relevance and freshness judgements on a random sample of 76 query URL pairs among three judgers are 0.88 and 1.02 respectively.

Freshness and relevance are evaluated by *hybrid NDCG*, and so when $\gamma = 0$ or $\gamma = 1$, this corresponds to NDCF [13] and NDCG, respectively.

Ranking features. The features used by RankSVM for ranking can be grouped into non-temporal and temporal features. The non-temporal features (summarized in Table 1) include several commonly used text-similarity scores such as BM25 [25], and language

[8] `http://www.mturk.com`

[9] Admittedly, judging for freshness according to an arbitrary time in past could be a difficult task. However, the choice was dictated to us by the time span of our dataset.

Table 1: Non-temporal ranking features used by RankSVM in the CS-DAC framework and baseline methods. Body, title, heading and anchor-text fields are respectively represented by B, T, H and A.

Feature name	Feature description	Feature name	Feature Description
Okapi(B)	Okapi BM25 score [25] for body-text.	RQT(B)	Ratio of covered terms in body-text.
RQT(H)	Ratio of covered terms in heading-text.	LM.JM(B)	body-text language modeling (Jelinek-Mercer) score [31].
LM.Dir(B)	Body-text language modeling (Dirichlet) score [31].	RQT(T)	Ratio of covered terms in title-text.
InNum	Number of inlinks.	TF(B)	Term frequency in body-text.
AvgNTF(B)	Average normalized TF in body-text.	LM.JM(T)	title-text language modeling (Jelinek-Mercer) score
STFIDF(H)	Sum of term TFIDF in heading-text.	NumQT(A)	Number of covered terms in anchor-text.
MaxNTF(B)	Maximum normalized TF in body-text.	PR	PageRank score [4].
AvgNTF(T)	Avgerage normalized TF for title-text.	LM.Dir(T)	title-text language modeling (Dirichlet) score.
MxTFIDF(T)	Maximum term TFIDF in title-text.	MaxNTF(T)	Maximum normalized TF in title-text.
LM.Dir(H)	heading-text language modeling (Dirichlet) score	MaxTF(T)	Maximum query term frequency in title-text.
ATFIDF(T)	Average term TFIDF in title-text.	AvgTF(T)	Average query term frequency in title-text.
SumTF(T)	Sum of term frequency in title-text.	LM.JM(H)	heading-text language modeling (Jelinek-Mercer) score.
L(B)	Body-text length.	AvgTF(H)	Average query term frequency in heading-text.
SumTF(H)	Sum of term frequency in heading-text.		

Table 2: Relevance and freshness judging guidelines for mechanical turk editors.

1. Relevance Evaluation.
Imagine you searched for "Mechanical Turk" in Google and got back a list of URLs in your results.

- A result of "www.mturk.com" would be a **highly relevant** match.
- A blog entry or news about working on Mechanical Turk would be **relevant**.
- A story about a person's daily life in which Mechanical Turk is mentioned in one sentence is treated as **borderline**.
- A story about an airplane in Turkey having had mechanical problems shortly after take off is **not relevant**.
- A story about a child eating fruits is considered **not related**.

2. Freshness Evaluation.
Use your knowledge about the query, combined with the time clues on the web page, including the time that the author wrote the story, the timestamp in copyright areas, etc., to judge whether the page is fresh or not, suppose you are in around April 2007.
Imagine you searched for "2007 cricket world cup" in Google around April 2007 and got back a list of URLs in your results.

- A news reporting the story of 2007 cricket world cup on previous one day would be **very fres**h.
- A critique about the fact that the ireland cricket coach is murdered in April 2007 is **fresh**.
- An introduction about the preparation of ireland cricket team for the world cup written in September 2006 is treated as **borderline**.
- A comment about stories in 2003 cricket world cup written in 2004 is **stale**.
- The introduction about the schedule of 2003 cricket world cup is **very stale**.

modeling [31], computed over different fields of documents (heading, title, body). The list also includes a few well-known link-based static features such as the number of inlinks and PageRank [4].

The temporal ranking features are generated by measuring the changes in the contents of documents with respect to their previous snapshots. For this purpose, we build a time series of each document's content changes, by going through the entire time span and comparing the TFIDF similarity of the document at each point with the previous and next versions. We generate separate time series for different document fields (heading, title, body), and use *STL seasonal-trend decomposition* [7] to decompose each time series τ into trend ($\mathcal{T}$), seasonal ($\mathcal{S}$) and remainder ($\mathcal{R}$) components.

$$STL(\tau) = \mathcal{T}_\tau + \mathcal{S}_\tau + \mathcal{R}_\tau$$

The same steps are repeated to decompose the time series generated based on link and page activities (create, remove, update) [8]. Figure 1 depicts an example of STL decomposition on a time series. In this instance, the time series (data) is generated from the frequency distribution of the query *jingle bells* in the logs of a commercial search engine. The same decomposition can be applied to a sequence of TFIDF scores, PageRank values or any other type of time series data. We use the output of STL decomposition for different time series to generate our temporal ranking features as summarized in Table 3. The slope of τ captures the speed of content changes, and has been suggested to be an effective feature for ranking [9]. The amplitude feature can measure the scale of content changes, and the position feature $Rp(\tau)$ is calculated with respect to the distance to the nearest peak in the time series. The confidence features are computed according to the distribution of $\mathcal{S}_\tau$ and $\mathcal{T}_\tau$ values after decomposition. We also employ the *Timed PageRank* of Yu et al. [30] as our temporally-sensitive static-rank feature.

Query clustering features. The query importance $\mathcal{I}$ features are used to cluster queries and assign the weights in each corresponding ranking function. We follow the approach taken by Bian et al. [2] and used the η top-ranked documents returned by a reference ranker (BM25 [25]) to generate our clustering features. We set the value of η to 15 in all our experiments. Once the pseudo-feedback documents are gathered, we compute the average value of each *ranking feature* over them and use the final mean value as a clustering feature. The feature importance is computed by training a reference RankSVM model for hybrid NDCG ($\gamma = 0.5$) on the training dataset.

Baseline methods. We compare the effectiveness of our CS-DAC with four baselines:

- Single ranker (SinR).
- Separate ranker training and selection (SepR).
- Over-weighting model [12].
- TopicalSVM [2].

In SinR, we train a single RankSVM ranker with all features. This could be regarded as a *weak* baseline that has no form of query categorization, and has been shown to perform more poorly than the

Table 3: Temporal ranking features used by RankSVM in the CS-DAC framework and baseline methods. The features (except for TPR) are produced from the STL decomposition [7] of time series generated from the content changes in title, body, heading, anchor, and page/link activities [8].

Feature name	Feature description
Slp(τ)	Slope of trend component T_τ.
Amp(τ)	Amplitude of seasonal component S_τ.
Rp(τ)	Relative position in S_τ.
Cs(τ)	Confidence of seasonality.
Cr(τ)	Confidence of regularity.
TPR	Timed PageRank [30].

other baselines in previous work [2, 16]. Nevertheless, we report its results because it represents one of the most common learning to rank architectures.

The SepR baseline is representative for the family of query-dependent loss function methods [2, 3, 16], in which the loss function is determined according to the temporal aspect of the query. Separate RankSVM rankers are trained for temporal and non-temporal queries, and each query is tested on the *correct* ranker for its type. Note that using the correct query type information—which is generally unavailable without manual effort—means that the performance numbers for this baseline are unaffected by potential query type misclassification, and therefore are overstated.

Dong et al. [12] investigated several techniques for ranking optimization with imbalanced amount of training data for freshness and relevance. Among their methods the *over-weighting* approach was most effective. The over-weighting model combines relevance and freshness labeled data to train a single ranker. This is similar to SepR except that the training pairs of the criterion with fewer labels are over-weighted. Dong et al. [12] used GBrank [32] as their ranking model. However, we modify the over-weighting loss function to RankSVM for consistency with the other methods in our experiments as follows:

$$\arg \min_{\omega, \xi_{q,i,j}} \frac{1}{2}\|\omega\|^2 + C \sum_{q,i,j} \xi_{q,i,j} \quad \textit{subject to}$$

$$\forall y_i^q \succeq y_j^q : \begin{cases} \frac{\alpha}{N_T}\omega^T X_i^q \geq \frac{\alpha}{N_T}\omega^T X_j^q + 1 - \xi_{q,i,j} & q \in \mathcal{Q}_T \\ \frac{1-\alpha}{N_N}\omega^T X_i^q \geq \frac{1-\alpha}{N_N}\omega^T X_j^q + 1 - \xi_{q,i,j} & q \in \mathcal{Q}_N \end{cases}$$

$$\forall_q \forall_i \forall_j : \quad \xi_{q,i,j} \geq 0$$

where $\mathcal{Q}_T$ and $\mathcal{Q}_N$ denote the sets of queries from Google Trends and MSN query log. N_T and N_N are respectively the number of preferential pairs of query-documents in each of those sets. α is a parameter that controls the balance of Google Trends queries vs. MSN queries, ranging over [0,1]. ω represents the feature weights within the ranking model.

Our last experimental baseline is TopicalSVM [2] which is the state-of-the-art in the family of divide and conquer techniques. TopicalSVM trains all rankers using a global loss function, and does not factorize the query-document importance $\mathcal{U}$ in contrast to CS-DAC.

5. EXPERIMENTS

We start our experiments by investigating the performance of our baseline techniques optimized for different goals. We then pick the best-performing baselines and compare them against CS-DAC. In all our experiments we run 5-fold cross-validation in which the first three folds are used for training, and the remaining two folds are used for validation and testing. The number of ranking functions (clusters) in CS-DAC and TopicalSVM to are set to three ($k = 3$), since preliminary results demonstrate CS-DAC and TopicalSVM perform the best when $k = 3$ and $k = 4$ (slightly outperforms the case when $k = 3$) respectively.

Table 4: Freshness comparison on the *temporal* (top) and *non-temporal* (bottom) query sets. All methods are trained using the hybrid labels and the evaluation is based on the freshness ratings ($\mathbf{y}^F$). Symbols †, §, and ‡ respectively denote statistically significant differences according to a single-tailed student t-test (p-value<0.05) over the SepR, TopicalSVM and Over-weighting baselines.

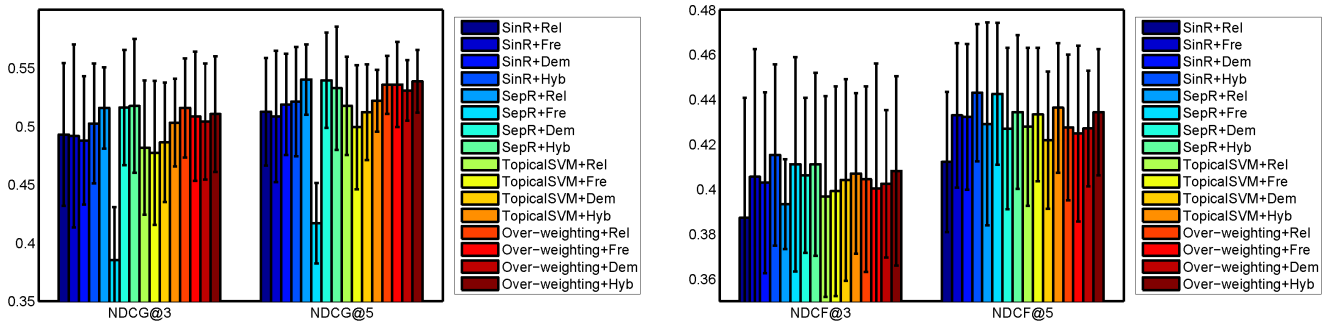
	Temporal Queries (Google Trends)			
	NDCF1	NDCF3	NDCF5	NDCF10
SepR	0.378	0.360	0.372	0.408
TopicalSVM	0.365	0.355	0.365	0.402
Over-weighting	0.340	0.348	0.363	0.404
CS-DAC	0.398‡	0.364	0.376	<u>0.411</u>
CS-DAC($\mathcal{U}$)	<u>0.416</u>†§‡	<u>0.379</u>‡	<u>0.388</u>	0.400
	Non-Temporal Queries (MSN logs)			
	NDCF1	NDCF3	NDCF5	NDCF10
SepR	0.348	0.411	0.434	0.475
TopicalSVM	0.355	0.408	0.430	0.485
Over-weighting	0.335	0.408	0.434	0.480
CS-DAC	0.427†§‡	0.454†§‡	0.473†§‡	0.510§‡
CS-DAC($\mathcal{U}$)	<u>0.452</u>†§‡	<u>0.466</u>†§‡	<u>0.488</u>†§‡	<u>0.527</u>†§‡

Performance analysis for experimental baselines. We investigate the performance of baseline techniques when trained for one of four optimization goals:

1. Relevance (Rel): The baselines are trained using relevance labels only.
2. Freshness (Fre): The baselines are trained using freshness labels only.
3. Hybrid labels (Hyb): The baselines are trained using hybrid labels (Equation 2).
4. Demoted labels (Dem): Dong et al. [12, 13] suggested *demoting* the the relevance grades of outdated documents. They suggested that if a document is somewhat outdated, then its relevance label should be demoted by one grade. For totally outdated documents the relevance labels are demoted by two grades. We followed the same strategy to compute our demoted labels. In essence, this is a special case of hybrid labeling.

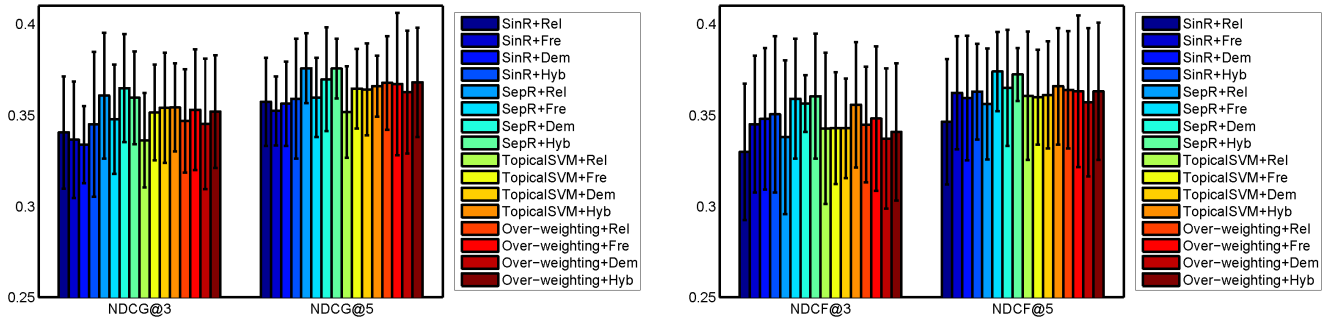
The final results of each optimized ranker are evaluated separately for freshness and relevance using NDCG with corresponding labels.

Figure 2 shows the performance of baseline techniques on the *non-temporal* query set (sampled from the MSN logs). As expected, when evaluating using the relevance labels ($\mathbf{y}^R$), it is more effective to optimize for relevance (Rel) rather than freshness (Fre). Similarly, optimizing for freshness produces results that have better NDCF values. The methods optimized for demoted (Dem) and hybrid (Hyb) labels consistently outperform those that are optimized for either freshness or relevance. The results also suggest that our hybrid labels are better for improving both relevance and freshness compared to the demoted labels of Dong et al. [12, 13].



(a) *Relevance labels* ($\mathbf{y}^R$) (b) *Freshness labels* ($\mathbf{y}^F$)

Figure 2: Ranking performance of baseline systems on relevance (left) and freshness (right) for the *non-temporal* query set. Error bars are the standard deviations of performance across five cross-validation folds.



(a) *Relevance labels* ($\mathbf{y}^R$) (b) *Freshness labels* ($\mathbf{y}^F$)

Figure 3: Ranking performance of baseline systems on relevance (left) and freshness (right) for the *temporal* query set. Error bars are the standard deviations of performance across five cross-validation folds.

Among the baselines, SinR has overall the poorest performance which is consistent with previous observations [3]. TopicalSVM, over-weighting and SepR show similar effectiveness while the latter might be considered marginally better—not surprising given that we use correct query type information in SepR.

We repeat the analysis on the *temporal* query set and the results are illustrated in Figure 3; as in the previous experiment, SinR has the lowest performance on both sets of labels while the other methods show similar effectiveness. Compared to the experiments on the non-temporal query set, there is less variation in performance when optimized for different types of labels. Our investigations revealed that this is due to high correlation between relevance and freshness labels on the temporal set. The Pearson's correlation between relevance and freshness labels on the temporal query set is 0.912 ± 0.004, statistically significantly higher than 0.429 ± 0.021 for the non-temporal set.

Based on the summarized results, we choose hybrid labels for training rankers in our remaining experiments. We also drop SinR as it consistently showed inferior effectiveness compared to all other methods.

Comparative performance on freshness. We use NDCG with freshness $\mathbf{y}^F$ labels (NDCF [13]) to compare the performance of CS-DAC with the baselines on both temporal (Google Trends) and non-temporal (MSN logs) query sets. We report the results for CS-DAC in the presence and absence of the query-document importance factor ($\mathcal{U}$) described in Equations 3 and 5. We respectively refer to these two versions as CS-DAC($\mathcal{U}$) and CS-DAC.

Table 4 includes the NDCF results on both query sets. The over-weighting baseline performs worst than the other methods. This is not surprising given that over-weighting is originally designed for scenarios with imbalanced training data [12], and the fact that it does not leverage any type of query classification or clustering. Consistent with the observations in the previous section, SepR and TopicalSVM produce similar results on the temporal queries, while they are both outperformed by CS-DAC. Introducing the $\mathcal{U}$ factor leads to further improvements in performance particularly at higher cutoffs. On non-temporal queries, TopicalSVM and SepR and over-weighting show similar effectiveness while CS-DAC consistently outperforms all baselines significantly. It is interesting to observe that CS-DAC improvements over the baselines are larger on the non-temporal query set. This can be explained by two reasons: (1) the documents returned for temporal queries tend to be fresher on average than those returned for the non-temporal ones, and (2) the high correlation between relevance and freshness labels in this set leads to more effective learning by reducing impact of potential noise in clustering and hybrid labels.

Comparative performance on relevance. We run a similar analysis, and compare the NDCG values of different techniques as

Table 5: Relevance comparison on the *temporal* (top) and *non-temporal* (bottom) query sets. All methods are trained using the hybrid labels and the evaluation is based on the freshness ratings ($\mathbf{y}^R$). Symbols †, §, and ‡ respectively denote statistically significant differences according to a single-tailed student t-test (p-value<0.05) over the SepR, TopicalSVM and Over-weighting baselines.

	Temporal Queries (Google Trends)			
	NDCG1	NDCG3	NDCG5	NDCG10
SepR	0.373	0.359	0.375	0.411
TopicalSVM	0.342	0.354	0.365	0.408
Over-weighting	0.355	0.351	0.368	0.411
CS-DAC	0.385	0.365	0.377	0.417
CS-DAC($\mathcal{U}$)	0.401†‡	0.375	0.389	0.426†
	Non-Temporal Queries (MSN logs)			
	NDCG1	NDCG3	NDCG5	NDCG10
SepR	0.481	0.517	0.532	0.562
TopicalSVM	0.490	0.508	0.521	0.566
Over-weighting	0.476	0.510	0.538	0.570
CS-DAC	0.493	0.520	0.541	0.574
CS-DAC($\mathcal{U}$)	0.509	0.522	0.541	0.574

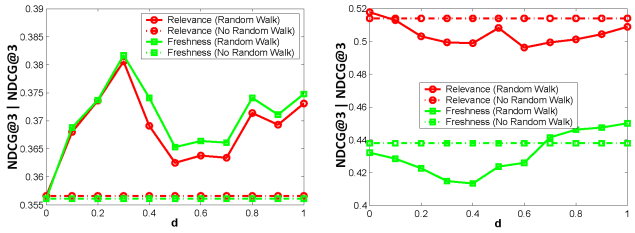
measured by the relevance labels ($\mathbf{y}^R$) in Table 5. For non-temporal queries, the CS-DAC results are marginally better than the baselines, although none of the differences are statistically significant. On the temporal query set, SepR has the edge over the other baselines while CS-DAC outperforms the three of them at all cutoff values. Adding the $\mathcal{U}$ factor significantly improves the results for NDCG@1 and NDCG@10. As in the NDCF numbers on this query set, the NDCG values could be also affected by the high correlation between freshness and relevance.

6. DISCUSSION

We showed that our CS-DAC method could significantly improve both freshness and relevance of the results compared to state-of-the-art baselines. In this section, we investigate the impact of random walk smoothing in improving the query-document factor $\mathcal{U}$ for training. We also compare CS-DAC and the baselines in terms of hybrid NDCG by assigning various weights to relevance and freshness. Finally, we report the most effective features according to our experiments for ranking temporal and non-temporal queries.

Smoothing query-document importance. We described earlier how original query-document importance values can be smoothed by random walk, where the probability d of random jumping can be tuned during training and validation. Figure 4 shows how choosing different fixed values for d may affect the results. On the non-temporal query set, different degrees of smoothing have little advantage over no smoothing ($d = 0$). On the temporal query set however, random-walk helps to smooth inter-label dependencies, and hence improves the results on both freshness and relevance.

Hybrid labels for evaluation. In Section 5 we showed that training for hybrid NDCG ($\gamma = 0.5$) was effective for improving both freshness and relevance. Here, we provide the evaluation results on hybrid NDCG, the metric we used for optimizing the ensemble ranking. Although we used $\gamma = 0.5$ for training, we report the evaluation results for different values of γ in Figure 5 to account for scenarios where freshness and relevance are



(a) *Temporal queries* (b) *Non-temporal queries*

Figure 4: The impact of changing the random jump probability d during smoothing of the query-document importance values $\mathcal{U}$. The results are evaluated on temporal (left) and non-temporal (right) queries using both relevance and freshness labels.

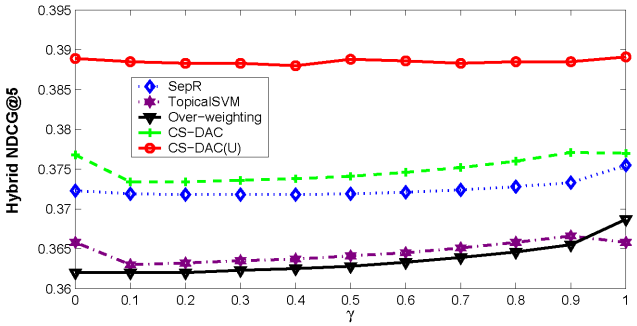
weighted differently. The results are consistent with our previous experiments; CS-DAC outperforms the baselines, and the weighting between freshness and relevance is less important for temporal queries. Increasing the γ value grows the overall hybrid NDCG almost monotonically because the relevance-based NDCG values are generally greater than those computed based on the freshness labels. It is worthwhile of pointing out that this observation does not suggest that ranking performance benefits the most when only optimizing for relevance.

Feature Analysis. CS-DAC relies on several temporal and non-temporal features for query clustering and document ranking. We examined all cross-validation folds to find the features that are assigned with highest weights during training. Among the temporal features, the confidence values for the seasonality CS(τ), and regularity Cr(τ) of STL decompositions were generally the most effective. Furthermore, the features generated from the time-series decomposition of changes in anchor-text and inlinks were more successful than those similarly produced based on other fields (e.g., title, body, heading). The effectiveness of temporal link-based features for improving relevance and freshness has been also acknowledged previously by Dai and Davison [8].

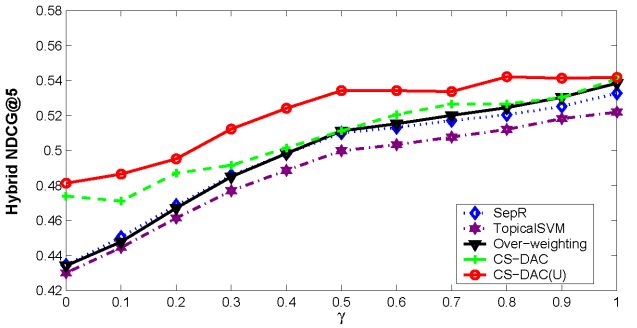
Between the non-temporal features, BM25 and language modeling scores had the highest weights and were most effective when computed over the body and title text.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a learning to rank approach (CS-DAC) for optimizing for relevance and freshness simultaneously. We extended the state-of-the-art in divide and conquer ranking [3] by adding two new key elements; first, instead of optimizing for relevance labels, we generated and used hybrid labels based on relevance and freshness grades. Second, we introduced a new query-document importance factor ($\mathcal{U}$) that allows each ranker to set different importance to relevance and freshness. Compared with traditional metasearch engines, divide-and-conquer ranking frameworks generate merged ranking lists on the model level instead of the result level. It enables automatic identification of effective ranking features for individual type of queries. Our experiments on a large web archive demonstrated that CS-DAC can improve both relevance and freshness compared to existing baselines. In the future, we plan to compare with data fusion methods, i.e, training two separate rankers with each utilizing judgments based on one criterion (relevance or freshness), and then merging results into a single ranking list. We also plan to consider the document diversity



(a) *Temporal queries*



(b) *Non-temporal queries*

Figure 5: Hybrid NDCG5 values for different values of γ on the *temporal* (top) and *non-temporal* (bottom) query set. Similar trends were found for NDCG at different cutoff values.

problem that is especially important for answering time-sensitive queries, e.g, a news-related query may have many news articles which may incur document duplication.

We studied the correlation between relevance and freshness grades, and its implications on the training effectiveness. Our results revealed high correlation between relevance and freshness labels in temporal queries, suggesting that the choice of document labels is less important for training on that set. We modeled document importance by the likelihood of visiting each unique hybrid label, and surprisingly found that it can improve ranking performance, especially for temporal queries. However, in what way that such document weighting strategies influence ranking performance is still unclear. We will leave it as one of our future work.

Our work can be considered as the simplest form of *multi-objective (multiple-criteria) optimization* [26], where multiple objective functions (freshness, relevance) are combined to form a single optimization goal (hybrid labels). These kinds of *aggregated* functions require the weight of each objective to be known in advance (γ in our case), and are incapable of finding all optimal solutions. Deploying more sophisticated multi-objective optimization techniques may lead to more significant improvements in relevance and freshness. Further work includes adopting other learning to rank architectures such as boosted decision trees [29] for multi-objective optimization of freshness and relevance.

Acknowledgments

We thank the anonymous reviewers for their useful comments. We also thank Jiang Bian for valuable discussion on TopicalSVM model and Jingjing Liu for helpful comments on the judgments for ranking evaluation. This work was supported in part by a grant from the National Science Foundation under award IIS-0803605.

8. REFERENCES

[1] K. Berberich, S. J. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *Proc. of ECIR*, pages 13–25, 2010.

[2] J. Bian, X. Li, F. Li, Z. Zheng, and H. Zha. Ranking specialization for web search: a divide-and-conquer approach by using topical RankSVM. In *Proc. of WWW*, pages 131–140, 2010.

[3] J. Bian, T.-Y. Liu, T. Qin, and H. Zha. Ranking with query-dependent loss for web search. In *Proc. of WSDM*, pages 141–150, 2010.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proc. of WWW*, pages 107–117, Apr. 1998.

[5] C. Burges, T. Shaked, E. Renshaw, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of ICML*, pages 89–96, 2005.

[6] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proc. of WWW*, pages 2–11, 2005.

[7] R. B. Cleveland, W. S. Cleveland, J. E. Mcrae, and I. Terpenning. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[8] N. Dai and B. D. Davison. Freshness matters: In flowers, food, and web authority. In *Proc. of SIGIR*, pages 114–121, 2010.

[9] N. Dai and B. D. Davison. Mining anchor text trends for retrieval. In *Proc. of ECIR*, pages 127–139, 2010.

[10] W. Dakka, L. Gravano, and P. G. Ipeirotis. Answering general time sensitive queries. In *Proc. of CIKM*, pages 1437–1438, 2008.

[11] F. Diaz. Integration of news content into web results. In *Proc. of WSDM*, pages 182–191, 2009.

[12] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proc. of WSDM*, pages 11–20, 2010.

[13] A. Dong, R. Zhang, P. Kolari, J. Bai, F. Diaz, Y. Chang, Z. Zheng, and H. Zha. Time is of the essence: improving recency ranking using twitter data. In *Proc. of WWW*, pages 331–340, 2010.

[14] J. L. Elsas and S. T. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proc. of WSDM*, pages 1–10, 2010.

[15] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, December 2003.

[16] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *Proc. of SIGIR*, pages 115–122, 2008.

[17] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *Proc. of WWW*, pages 381–390, 2009.

[18] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proc. of SIGIR*, pages 41–48, 2000.

[19] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of SIGKDD*, pages 133–142, 2002.

[20] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Transactions on Information Systems*, 25(3):14, 2007.

[21] I. Kang and G. Kim. Query type classification for web document retrieval. In *Proc. of SIGIR*, pages 64–71, 2003.

[22] A. Kulkarni, J. Teevan, K. Svore, and S. Dumais. Understanding temporal query dynamics. In *Proc. of WSDM*, pages 167–176, 2011.

[23] L. Li, F. Liu, and W. Chou. An information theoretic approach for using word cluster information in natural language call routing. Technical Report ALR-2003-014, Avaya Labs Research, 2003.

[24] D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proc. of SIGIR*, pages 700–701, 2009.

[25] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proc. of CIKM*, pages 42–49, 2004.

[26] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, 546 pp, 1986.

[27] R. Tibshirani, G. Walther, and T. Hastie. Estimating the Number of Clusters in a Dataset via the Gap Statistic. *Journal of the Royal Statistical Society, Series B*, 63:411–423, 2000.

[28] L. Wang, J. Lin, and D. Metzler. Learning to efficiently rank. In *Proc. of SIGIR*, pages 138–145, 2010.

[29] Q. Wu, C. Burges, K. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13, 2010.

[30] P. S. Yu, X. Li, and B. Liu. On the temporal dimension of search. In *Proc. of WWW*, pages 448–449, 2004.

[31] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22:179–214, 2004.

[32] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in NIPS 20*, 2008.