Joint Relevance and Freshness Learning From Clickthroughs for News Search

Hongning Wang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana IL, 61801 USA
wang296@illinois.edu

Anlei Dong, Lihong Li, Yi Chang,
Evgeniy Gabrilovich
Yahoo! Labs
701 First Avenue, Sunnyvale, CA 94089
{anlei,lihong,yichang,gabr}@yahoo-inc.com

ABSTRACT

In contrast to traditional Web search, where *topical relevance* is often the main selection criterion, news search is characterized by the increased importance of *freshness*. However, the estimation of relevance and freshness, and especially the relative importance of these two aspects, are highly specific to the query and time the query was issued. In this work, we propose a unified framework for modeling the topical relevance and freshness, as well as their relative importance, based on click logs. We use click statistics and content analysis techniques to define a set of temporal features, which predict the right mix of freshness and relevance for a given query. Experimental results on both historical click data and editorial judgments demonstrate the effectiveness of the proposed approach.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Relevance and freshness modeling, temporal features

1. INTRODUCTION

When a user submits the query "Apple Company" into a news search engine, she is expecting to find a list of most recent news reports that are topically relevant to the company. The emphasis on recency is crucial, as even the seemingly current events can quickly become outdated, dwarfed by the importance of new developments. For example, the news articles covering the release of iPhone 4S were quite relevant on Oct 4, 2011; however, they became less relevant just one day later, when Apple Inc.'s former CEO Steve Jobs passed away. Such cases are common in news search and make it different from the traditional web search, where "relevance" is typically narrowly defined as topical relatedness. In web search, a great amount of effort has been devoted to designing effective retrieval features and models to improve the estimation of topical relevance [22, 15, 4, 11, 20]; however, for news search much less work has been done to take freshness into account. The importance of content recency in news search suggests extending the conventional notion of relevance beyond pure topical match, by incorporating freshness as another important ranking criterion.[1]

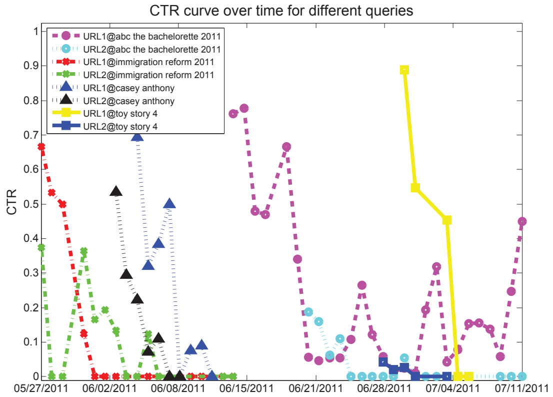


Figure 1: CTR curves over 8 different URLs over a two-month period.

Incorporating freshness into news ranking is not a trivial exercise of combining relevance and freshness scores, as users' search intents need to be taken into consideration. For breaking news queries, as in our previous example, returning most recent news would always improve users' satisfaction. In other cases, for newsworthy queries such as "bin laden death", preferring older but more relevant reports, in terms of coverage and authority, makes more sense. Previous studies have shown that the relationship between returned documents and queries varies as users' intent in web search changes over time [17, 19]. To demonstrate this phenomenon in news search scenario, we select 8 most frequently returned URLs under 4 different queries (2 URLs per query) from Yahoo! news search engine[2] in a period of two months (late May to late July, 2011). Two of these queries have the highest search frequency within one day, namely, "toy story 4" and "casey antony", and the other two have the longest lifetime span over this period, i.e., "abc the bachelorette 2011"

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

[1] In what follows, we use the term "relevance" to only refer to topical relatedness to avoid ambiguity.
[2] `http://news.search.yahoo.com/`

and "immigration reform 2011". We illustrate the Click-Through Rate (CTR) curves for these 8 URLs during the above period in Figure 1 and list the corresponding URLs in Table 1.

Table 1: URL list for Figure 1

Entry	URL
URL1@ abc the bachelorette 2011	http://blog.zap2it.com/frominsidethebox/2011/06/tv-ratings-bachelorette-leads-abc-again-monday-stanley-cup-numbers-rise.html
URL2@ abc the bachelorette 2011	http://www.broadcastingcable.com/article/470053-Primetime-Ratings-Bachelorette-Drops-ABC-Still-Wins-Monday.php?rssid=20065
URL1@ immigration reform 2011	http://biz.yahoo.com/prnews/110526/la10138.html?.v=1
URL2@ immigration reform 2011	http://seattletimes.nwsource.com/html/editorials/2015089638-edit19dream.html?syndication=rss
URL1@ casey anthony	http://www.cnn.com/2011/CRIME/06/04/casey.anthony.weekly.wrap/index.html?section=cnn-latest
URL2@ casey anthony	http://www.cnn.com/2011/CRIME/06/03/florida.casey.anthony.trial/index.html?eref=rss-us
URL1@ toy story 4	http://theenvelope.latimes.com/news/la-et-at-toy-story-4-oscar-sl,0,1068280.storylink?track=rss
URL2@ toy story 4	http://1019litefm.radio.com/2011/06/28/toy-story-4-might-happen/

From the CTR curves, we can clearly observe the difference between these two types of queries. For the query "casey antony", the CTR for the returned URLs quickly diminished because new stories came out soon. On the other hand, for the query "abc the bachelorette 2011", users' interest was maintained over much longer periods of time. In the latter case, the URLs contained summary report, and users found them useful long after this TV episode was shown. These results imply that users' preferences are highly dependent on the query as well as its issuing time.

Given the highly dynamic nature of news events and the sheer scale of news reported around the globe, it is often impractical for human editors to constantly keep track of the news and provide timely relevance and freshness judgments. Since machine learned ranking methods often depend on editorial judgments [20], delayed and inaccurate annotations can mislead the learning algorithms. To this end, Dong et al. [8] designed a set of crawling mechanisms and editorial guidance to annotate the relevance and freshness. Then they used the freshness grade to demote the relevance grade when computing the final ranking score. However, we believe such rule-based demotion is often sub-optimal. To validate this conjecture, we asked editors to annotate the news query log for the day of Aug. 9, 2011 immediately one day after, and then demoted the news search results based on their judgments using the algorithm of Dong et al. [8]. The relation between observed CTR and the demoted grades is visualized by a scatter plot in Figure 2.

From Figure 2, we see that the clicks are not strictly correlated with the demoted grades: the average Pearson correlation between them across the queries is 0.5764 with standard deviation 0.6401. The main reason for this inconsistency is the hard demotion rule: users might have different demotion preferences for different queries, and it's almost impossible for an editor to predefine the rules given the plurality of possibilities. As a result, the uncertainty from this heuristically-derived ranking grades will limit the performance of subsequent learning-to-rank algorithms.

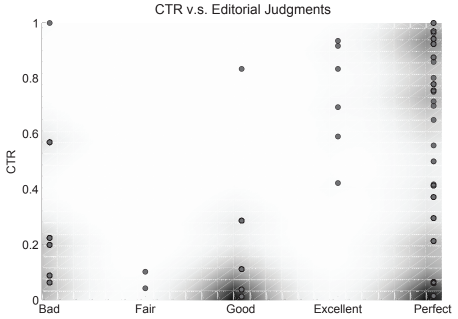


Figure 2: Scatter plot of CTR versus editor's relevance judgments

In this work, we propose to model relevance and freshness, and the query-specific relative mix of these two aspects simultaneously from the click logs. We assume the users' click behavior for the given query depends jointly on both the relevance and freshness of a news article. To this end, we introduce a latent factor model, Joint Relevance Freshness Learning (JRFL), which captures the relevance and freshness aspects, as well as the latent preference between the two, in a unified way. To capture the temporal sensitivity of both news documents and queries, a set of effective temporal features, utilizing click statistics and content analysis techniques, is proposed. We evaluate the proposed method on both click data and editorially annotated sessions. Experimental evaluation confirms that the proposed learning method outperforms several standard learning-to-rank algorithms, which cannot properly handle the query-specific trade-off between relevance and freshness.

The contributions of our joint learning method are two folds:

1. The relevance and freshness are jointly learned from the click logs, which avoids defining any hard combination rules for relevance and freshness ahead of time , and such query-specific preference is directly estimated from the data.

2. Our method does not require any manually annotated data, making it applicable in a broad spectrum of retrieval tasks, where task-specific ranking criteria can be easily incorporated.

2. RELATED WORK

Learning-to-rank algorithms have shown significant and consistent success in various applications [20, 13, 25, 5]. Such machine-learned ranking algorithms learn a ranking mechanism by optimizing particular loss functions based on editorial annotations. An important assumption in those learning methods is that the "relevance" of documents for a given query is generally stationary over time, so that, as long as the coverage of the labeled data is broad enough, the learned ranking functions would generalize well to future unseen data. Such assumption is often true in web search, but less likely to hold in news search because of the dynamic nature of news event and lack of timely annotations, as we have analyzed in Section 1.

Recency ranking is an emerging research topic to tackle the time-sensitive ranking problems in web search. Li et al. [19] and Efron et al. [10] came up with solutions from a content analysis perspective, by introducing document publication timestamp into language models. However, the temporal property for both queries and documents are not limited to timestamps alone; various signals are available to depict it. Dong et al. [8] designed a system to automatically detect and response to recency sensitive queries. Later on, features extracted from Twitter[3] stream were incorporated to identify fresh URLs [9]. But their learning method depended on a predefined freshness-demotion strategy, which does not necessarily lead to optimal generalization performance. In another effort by Moon et al. [21], user clicks were combined with a baseline ranking system to capture temporal shifts of a user's information need in recency search. However, they only used clicks for the highest ranked article to update their ranking model, thus did not make full use of click data.

The closet work to ours is Dai et al.'s divide-and-conquer learning strategy for recency ranking [7], although their work was substantially different from ours. First, they still relied on the manual relevance/freshness annotations to train the rankers, where a weighted harmonic mean was used to integrate the relevance and freshness labels. In JRFL, we do not require such manual annotations — relevance and freshness are automatically learned from the clickthroughs, resulting in greater flexibility in our model. Second, they did not model relevance and freshness separately, but instead used one ranker on all the features. In our method, we learn the relevance and freshness models *separately* from two different sets of features. The separation allows the two models to focus better on different aspects of a document (namely, freshness and relevance). Third, the importance weights between relevance and freshness were manually tuned in their harmonic mean, while our method uses a set of query-specific features to adaptively combine freshness and relevance, and the adaptation is automatically learned from real clicks.

Since we are learning to optimize different ranking criteria, our work is also related to multi-object ranking. Svore et al. proposed to optimize multiple graded ranking measures, e.g., NDCG and CTR, by combining the gradients from different object functions in the framework of LambdaMART [23]. Agarwal et al. used a constrained optimization framework to encode multiple objectives for clicks and post-click downstream utilities in content recommendation systems [1]. The problem tackled by JRFL is somewhat different: we still try to optimize the information utility of a ranking list, although the utility is not directly observed and is affected by two criteria (relevance and freshness) in an unknown and query-specific way. Our goal is to simultaneously learn, from click data, the two criteria as well as the best combination of them.

3. METHOD

Suppose, when a user submits a query to a news search engine and gets an according list of ranked news documents, she would first judge the usefulness of each document by her underlining sense of relevance and freshness, and gives it an overall impression grade by her preference over relevance and freshness at that particular time. Once she has such impressions in mind, she would deliberately click the documents most interesting to her and skip all the others.

Inspired by this example, we proposed to model the users' click behavior in news search as a direct consequence of examining the relevance and freshness aspects of the returned documents. Besides, for different queries, the relative emphasis the users put over these two aspects can vary substantially, reflecting the searching intention for the specific news event. Therefore, a good ranking function should be able to infer such a trade-off and return the "*optimally combined*" ranking results for individual queries. However, we cannot explicitly obtain the users' relevance/freshness judgment and the preferences over these two aspects, since their evaluation process is not directly observable from the search engine. Fortunately, the users' click patterns are recorded, from which we can assume the clicked documents are more meaningful to her than the non-clicked ones [14]. Therefore, we model relevance and freshness as two latent factors and assume a linear combination of these two, which is also latent, generates the observed click preferences.

To better determine the temporal property of the news documents and detect the recency preference imposed for the query, we design a set of novel temporal features from click statistics and content-analysis techniques. In the following sections, we will introduce the proposed model and temporal features in detail.

3.1 Joint Relevance and Freshness Learning

The basic assumption of our proposed *Joint Relevance Freshness Learning (JRFL)* model is that a user's overall impression assessment by combining relevance and freshness for the clicked URLs should be higher than the non-clicked ones, and such a combination is specific to the issued query. Therefore, our method falls in to the pairwise-learning-to-rank framework.

Formally, we have N different queries and for the n-th query we observed M different URL click preference pairs $(U_{ni} \succ U_{nj})$, in which U_{ni} is clicked but U_{nj} is not. We denote X^R_{ni} and X^F_{ni} as the relevance and freshness features for U_{ni} under Query Q_n, and S^R_{ni} and S^F_{ni} are the corresponding relevance score and freshness score for this URL given by the relevance model $g_R(X^R_{ni})$ and freshness model $g_F(X^F_{ni})$, respectively. In addition, we denote α^Q_n as the relative emphasis on freshness estimated by the query model $f_Q(X^Q_n)$, i.e., $\alpha^Q_n = f_Q(X^Q_n)$, based on the features X^Q_n describing query Q_n. To make relevance/freshness scores comparable across all the URLs, we require $0 \leq \alpha^Q_n \leq 1$. As a result, the user's latent assessment Y_{ni} about the URL U_{ni} for a particular query Q_n is assumed to be a linear combination of its relevance and freshness scores:

$$Y_{ni} = \alpha^Q_n \times S^F_{ni} + (1 - \alpha^Q_n) \times S^R_{ni} \qquad (1)$$

Since we have observed the click preference $(U_{ni} \succ U_{nj})$, we can safely conclude that $Y_{ni} > Y_{nj}$.

Based on the previous discussion, we characterize the observed pairwise click preferences as the joint consequences of examining the combined relevance and freshness aspects of the URLs for the query. For a given collection of click logs, we are looking for a set of optimal models (g_R, g_F, f_Q), which can explain the observed pairwise preferences as many as possible. As a result, we formalize this pairwise

[3] http://twitter.com/

learning problem as an optimization task,

$$\min_{f_Q,\ g_R,\ g_F,\ \xi} \ \frac{1}{2}(\|f_Q\|+\|g_R\|+\|g_F\|)+\frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij} \quad (2)$$

$$\begin{aligned} \text{s.t.}\quad & \forall(n,i,j), URL_{ni} \succ URL_{nj} \quad (3)\\ & Y_{ni}-Y_{nj} > 1-\xi_{nij}\\ & 0 \le f_Q(X_n^Q) \le 1\\ & \xi_{nij} \ge 0, \end{aligned}$$

where Y_{ni} and Y_{nj} are defined in Eq (1), the non-negative slack variables $\{\xi_{nij}\}$ are introduced to account for noise in the clicks, $\|\cdot\|$ is functional norm (to be defined later) describing complexity of the models, and C is the trade-off parameter between model complexity and training error.

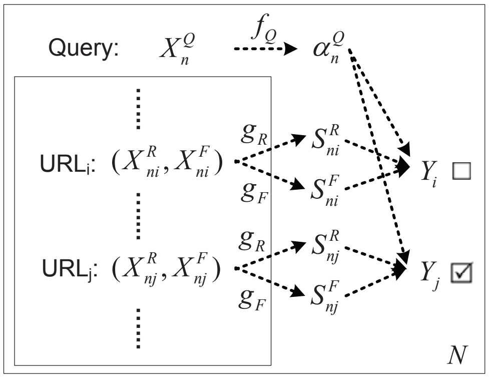


Figure 3: Intuitive illustration of the proposed Joint Relevance and Freshness Learning model. The user issued query and the corresponding returned URLs are represented by their features on the left part. Dashed arrow lines in the middle indicate the assumed user's judging process before she clicks. The check boxes on right record the clicked URLs.

Figure 3 depicts the intuition behind JRFL. From the figure, we can clearly notice the difference between our proposed JRFL and other classic pairwise-learning-to-rank algorithms, e.g., RankSVM [13] and GBrank [25]. A classic pairwise-learning-to-rank algorithm only uses one scoring function to account for all the observed click preferences, where different ranking criteria cannot be easily incorporated. Besides, even though RankSVM model shares a similar object function as JRFL, they are still quite different: JRFL simultaneously learns a relevance model and a freshness model, and utilizes a query-specific model to leverage these two aspects together to explain the observed click patterns. Neither RankSVM nor GBRank deal with such query-specific multi-criterion ranking. Besides, as we have discussed in Section 2, in previous studies [8, 7], S_{ni}^R and S_{ni}^F for each URL are already known, and they tune α_n^Q directly for each type of query. In our method, all those factors are latent and estimated automatically from the click logs.

In our previous description, we didn't specify the forms of the relevance model $g_R(X^R)$, freshness model $g_F(X^F)$, and query model $f_Q(X^Q)$. Although many alternatives exist, we choose linear functions for all these models to simplify the exposition and derive efficient model estimation procedures. Other types of functions can also be employed, although numerical approximation is unavoidable in general when optimizing their model parameters.

Formally, we use three linear models:

$$g_R(X_{ni}^R) = w_R^{\mathsf{T}} X_{ni}^R \quad (4)$$

$$g_F(X_{ni}^F) = w_F^{\mathsf{T}} X_{ni}^F \quad (5)$$

$$f_Q(X_n^Q) = w_Q^{\mathsf{T}} X_n^Q \quad (6)$$

where the bias factor b in linear functions is excluded by introducing the dummy feature 1.

As a result, the proposed JRFL method defined in Eq(2) can be instantiated as:

$$\min_{w_R, w_F, w_Q, \xi} \frac{1}{2}(\|w_Q\|^2+\|w_R\|^2+\|w_F\|^2)+\frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij} \quad (7)$$

$$\begin{aligned} \text{s.t.}\quad & \forall(n,i,j), U_{ni} \succ U_{nj}\\ & w_Q^{\mathsf{T}} X_n^Q \times w_F^{\mathsf{T}}(X_{ni}^F - X_{nj}^F)\\ & \quad + (1-w_Q^{\mathsf{T}} X_n^Q) \times w_R^{\mathsf{T}}(X_{ni}^R - X_{nj}^R) > 1-\xi_{nij}\\ & 0 \le w_Q^{\mathsf{T}} X_n^Q \le 1\\ & \xi_{nij} \ge 0. \end{aligned}$$

Thanks to the associative property of linear functions, the optimization problem defined in Eq (7) can be divided into two sub-problems: relevance/freshness model estimation and query model estimation, and each of them is a convex program (note that Eq (7) itself is non-convex). Therefore, we can utilize the coordinate descent algorithm to iteratively solve the two convex programs, as shown in Figure 4.

The interaction between the relevance/freshness models and query model is clearly stated in the optimization process: in relevance/freshness models estimation, the query-specific preference weight acts as a tuning factor, which increases or decreases the features in these two models to represent the searching intention; once we have the relevance/freshness models, we tune the query model to preserve more click preferences based on the current relevance/freshness prediction.

Since each update step is convex, our coordinate optimization is guaranteed to decrease the object function in Eq (7) monotonically and therefore converges to a local optimum.

3.2 Temporal Features

We use 95 basic text matching features, such as query term matched in title, matched position in document, and source authority score from a subset of features employed in Yahoo! news search engine, as our URL relevance features. To capture the temporal property of the news documents and queries, we propose a set of novel time-sensitive features as our URL freshness features and query features, which are summarized in Table 2.

3.2.1 URL Freshness Features

Publication age $\mathbf{age}_{\text{pubdate}}$(URL|Query): the URL's publication timestamp is used to identify the document's freshness.

However, for news search, the freshness of news content is more important. Therefore, we propose to identify the given news document's freshness quality by content analysis techniques.

Story age $\mathbf{age}_{\text{story}}$(URL|Query): we use the regular expressions defined in [8] to extract the mentioned dates in the news content, calculate their distances to the given query

Table 2: Temporal Features for URL freshness and Query model

Type	Feature
URL freshness	$\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query}) = \text{timestamp}(\text{Query}) - \text{pubdate}(\text{URL})$
	$\mathbf{age}_{\text{story}}(\text{URL}\mid\text{Query}) = \text{timestamp}(\text{Query}) - \text{pubdate}_{extracted}(\text{URL})$
	$\mathbf{LM@1}(\text{URL}\mid\text{Query}, t) = \max_{d\in\text{Corpus(q}\mid\text{t)}[t,t-1\text{day}]} \log p(\text{URL}\mid d)$
	$\mathbf{LM@5}(\text{URL}\mid\text{Query}, t) = \max_{d\in\text{Corpus(q}\mid\text{t)}[t-2\text{days},t-5\text{days}]} \log p(\text{URL}\mid d)$
	$\mathbf{LM@ALL}(\text{URL}\mid\text{Query}, t) = \max_{d\in\text{Corpus(q}\mid\text{t)}[t-6\text{days},-\infty]} \log p(\text{URL}\mid d)$
	$\mathbf{t\text{-}dist}(\text{URL}\mid\text{Query}) = \frac{\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})-\text{mean}[\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})]}{\text{dev}[\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})]}$
Query Model	$\mathbf{q\_prob}(\text{Query}\mid\text{t}) = \log \frac{Count(Query\mid t)+\delta_q}{\sum_q Count(Query\mid t)+\delta}$
	$\mathbf{u\_prob}(\text{User}\mid\text{t}) = \log \frac{Count(User\mid t)+\lambda_u}{\sum_q Count(User\mid t)+\lambda}$
	$\mathbf{q\_ratio}(\text{Query}\mid\text{t}) = \mathbf{q\_prob}(Query\mid t) - \mathbf{q\_prob}(Query\mid\text{t-1})$
	$\mathbf{u\_ratio}(\text{User}\mid\text{t}) = \mathbf{u\_prob}(User\mid t) - \mathbf{u\_prob}(User\mid\text{t-1})$
	$\mathbf{Ent}(\text{Query}\mid\text{t}) = -p(Query\mid t) \log p(Query\mid t)$
	$\mathbf{CTR}(\text{Query}\mid\text{t}) = \text{mean}\left[\text{CTR(URL}\mid\text{Query, t)}\right]$
	$\mathbf{pub\_mean}(\text{Query}\mid\text{d}) = mean_{URL\in Corpus(Q\mid t)}\left[\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})\right]$
	$\mathbf{pub\_dev}(\text{Query}\mid\text{d}) = dev_{URL\in Corpus(Q\mid t)}\left[\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})\right]$
	$\mathbf{pub\_frq}(\text{Query}\mid\text{t}) = \log \frac{Count(URL\mid d)+\sigma_u}{\sum_{URL} Count(URL\mid t)+\sigma}$

(δ_q, δ), (λ_u, λ) and (σ_u, σ) are the smoothing parameters estimated from the query log.

```
**Algorithm**: Coordinate Descent for JRFL
*Input*: A collection of click preferences $L = \Big\{\big[X_n^Q, \big((X_{ni}^R, X_{ni}^F) \succ (X_{nj}^R, X_{nj}^F)\big), \ldots, \big((X_{nk}^R, X_{nk}^F) \succ (X_{nl}^R, X_{nl}^F)\big)\big]\Big\}$;
*Input*: Trade-off parameter $C$;
*Input*: Maximum iteration step $S$;
*Input*: Relative convergency bound $\epsilon$;
*Output*: Learned model parameters of $(w_R, w_F, w_Q)$;
***Step 0***: Randomly initialize $(w_R, w_F, w_Q)$ and set $i = 0$;
***Step 1***: Update Relevance/Freshness models:
    $$(w_R^{(i+1)}, w_F^{(i+1)}) \leftarrow \arg\min_{w_R, w_F, \xi} \frac{1}{2}(\|w_R\|^2+\|w_F\|^2)+\frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij}$$
    with respect to the constrains listed in Eq (7) by fixing the Query model to $w_Q^{(i)}$;
***Step 2***: Update Query model:
    $$w_Q^{(i+1)} \leftarrow \arg\min_{w_Q, \xi} \frac{1}{2}\|w_Q\|^2 + \frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij}$$
    with respect to the constrains listed in Eq (7) by fixing the Relevance/Freshness models to $(w_R^{(i+1)}, w_F^{(i+1)})$;
***Step 3***: Compute object function value defined in Eq (7)→ *obj* and increase $i = i + 1$;
***Step 4***: If the relative change in *obj* is greater than $\epsilon$ and i is smaller than $S$, go to ***Step 1***, else return $(w_R^{(i)}, w_F^{(i)}, w_Q^{(i)})$ .
```

Figure 4: Coordinate Descent for JRFL.

within the document, and select the one with the minimal distance as the extracted story timestamp to infer the corresponding story age.

Story coverage $\mathbf{LM@\{1,5,ALL\}}(\text{URL}\mid\text{Query}, t)$: content coverage is an important character of the freshness of a news document. Newer stories should cover more content that has not been mentioned by the previous reports. For a given query with a list of candidate news articles at a particular time, we first collect all the previous news articles associated with this query in our query log, and build language models [16, 24] for each of these documents. Then, we separate those language models into three sets: models with documents published one day before, two to five days before, and all the rest, and treat the candidate URLs as query to calculate the maximum generation probability given by all those models in these three sets accordingly.

Relative age $\mathbf{t\text{-}dist}(\text{URL}\mid\text{Query})$: from a user's perspective, since most of news search engines have already displayed $\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})$ to her, the document's relative freshness within the returned list is more meaningful for her. To capture this signal, we shift each URL's $\mathbf{age}_{\text{pubdate}}(\text{URL}\mid\text{Query})$ value within the returned URL list by the mean value of *publication age* in this list and scale the results by the corresponding standard deviation.

3.2.2 Query Freshness Features

The query features are designed to capture the latent preference between relevance and freshness.

Query/User frequency $\mathbf{q\_prob}(\text{Query}\mid\text{t})$ and $\mathbf{u\_prob}(\text{User}\mid\text{t})$: the frequency of a query within a fixed time slot is a good indicator of whether this is a breaking news query. We calculate the frequency of the query and unique users who issued this query within in a time slot prior to the query time.

Frequency ratio $\mathbf{q\_ratio}(\text{Query}\mid\text{t})$ and $\mathbf{u\_ratio}(\text{User}\mid\text{t})$: the relative frequency ratio of a query within two consecutive time slots implies the change of users interest. A higher ratio indicates an increasing user interest on this news event.

Distribution entropy **Ent**(Query|t): the distribution of query's issuing time is a sign of breaking news: a burst of search occurs when particular news event happens. So we utilize the entropy of query issuing time distribution to capture such burstiness. A multinomial distribution $p(Query|t)$ with fixed bin size (e.g., 2 hours per bin) is employed to approximate the query's temporal distribution within the day.

Average CTR **CTR**(Query|t): CTR is another signal representing the freshness preference of the query: when breaking news happens, people tend to click more returned URLs. We calculate the average CTR over all the associated URLs within a fixed time slot in prior to the query time.

URL recency **pub_mean**(Query|d), **pub_dev**(Query|d) and **pub_frq**(Query|d): the recency of the URLs associated with the query can be treated as a good profile of this query's freshness tendency: when the URLs associated with one particular query in a fixed period are mostly fresh, it indicates the query itself is highly likely to be a breaking news query. We calculate the mean and standard deviation of the associated URLs' $\mathbf{age}_{\text{pubdate}}$(URL|Query) features and the frequency of the URLs created in that specific period.

4. EXPERIMENT RESULTS

This section validates our JRFL model empirically with large-scale click data sets and editorial annotations. We begin by describing the data sets used.

4.1 Data Sets

4.1.1 Click Data Sets

We collected real search sessions from Yahoo! news search engine in a two months period, from late May to late July, 2011. And to unbiasedly compare different ranking algorithms, we also set up a random bucket to collect exploration clicks from a small portion of traffic at the same time. In this random bucket, the top four URLs were randomly shuffled and displayed to the real users. By doing such random shuffling, we were able to collect user click feedback on each document without positional bias, and such feedback can be thought as a reliable proxy on relevance of documents [18]. As a result, we only collected the top 4 URLs from this random bucket.

In addition, we also asked editors to annotate the relevance and freshness in Aug. 9, 2011's query log immediately one day after, according to the editorial guidance given by Dong et al. [8].

Simple preprocessing is applied on these click data sets: 1) filtering out the sessions without clicks, since they were useless for either training or testing in our experiments; 2) discarding the URLs whose publication time was after the query's issuing time (caused by errors from news sources); 3) discarding sessions with less than 2 URLs.

4.1.2 Preference Pair Selection

We decide to train our model on the normal click data, because such clicks are easier to collect without hurting the system's performance. However, this kind of clicks are known to be heavily positional biased [2]. To reduce the bias for training, we followed Joachims et al.'s method to extract preferences from clicks [14]. In particular, we employed two click heuristics:

1. "*Click $\succ$ Skip Above*": For a ranked URL list $\{U_1, U_2, \ldots, U_m\}$ and a set C containing the clicked URLs, extract a preference pair $U_i \succ U_j$ for all pairs $1 \le j < i$ with $U_i \in C$ and $U_j \notin C$.

2. "*Click $\succ$ Skip Next*": For a ranked URL list $\{U_1, U_2, \ldots, U_m\}$, and a set C containing the clicked URLs, extract a preference pair $U_i \succ U_{i+1}$ for all $U_i \in C$ and $U_{i+1} \notin C$.

In addition, to filter out noisy and conflicting preferences, we defined three rules: 1) filter out the preference pairs appearing less than 5 times; 2) calculate Pearson's χ^2 value [6] on all the pairs, and order them according to their χ^2 value; 3) if both $\mathrm{U}_i \succ \mathrm{U}_j$ and $\mathrm{U}_j \succ \mathrm{U}_i$ are extracted, discard the one with smaller χ^2 value.

After these selection steps, we were able to keep the top 150K preference pairs from some portion of normal clicks we collected in Section 4.1.1. Besides, for the testing purpose, we randomly select 500k query-URL pairs from original normal click set (not including the URLs used for generating the click preference pairs) and 500k from the random bucket clicks. In order to guarantee the quality of freshness annotation, we asked the editors to finish the annotation in day. As a result, we only have about 13k query-URL pairs annotated out of that day's query log. As a summary, we list the data sets for our experiments in Table 3.

Table 3: Evaluation Corpus

	#(Q,t)	#(Q,U,t)	#URL Pairs
Training preferences	75,236	230,351	150,000
Normal clicks	59,062	500,000	-
Random clicks	127,474	500,000	-
Editorial judgment	1,404	13,091	-

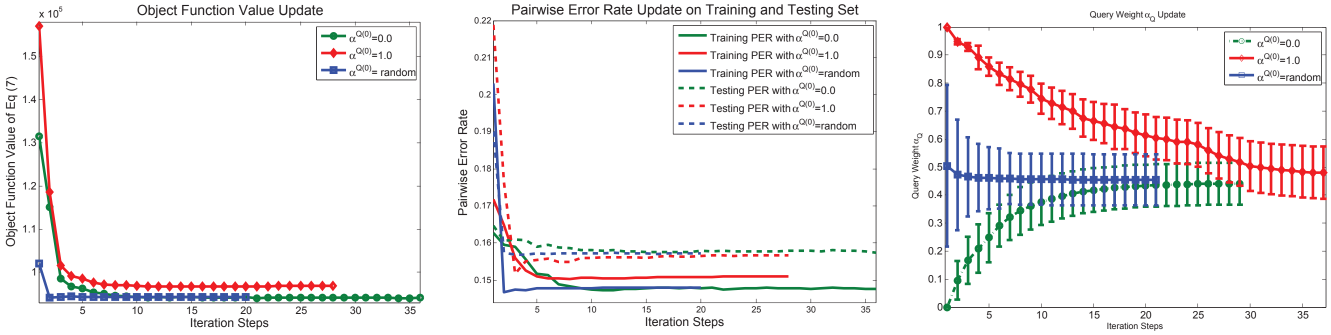
4.1.3 Temporal Feature Implementation

For most of our temporal features, we had to specify a time slot for the implementation; for example, the *Query/User frequency* **q_prob**(Query|t) and **u_prob**(User|t) are both calculated within a predefined time slot. In the following experiments, we set such a time slot to be 24 hours, and all the necessary statistics were collected from this time window accordingly. Once the features were generated, we linearly scaled each of them into the range [-1, 1] to normalize them.

4.1.4 Baselines and Evaluation Metrics

Since the proposed JRFL works in a pairwise-learning-to-rank manner, we employed two classic pairwise-learning-to-rank algorithms, RankSVM [13] and GBRank [25], as our baseline methods. Because these two algorithms do not explicitly model relevance and freshness aspects in ranking, we fed them with the concatenation of all our URL relevance/freshness and query features. Besides, to compare the models trained on clicks with those trained on editorial judgments, we also introduced Dong et al.'s freshness-demotion-trained GBRank model [8] as our baseline and denoted it as "FreshDem".

To quantitatively compare different ranking algorithms' retrieval performance, we employed a set of standard evaluation metrics in information retrieval. In click data, we treat all the clicked URLs as relevant and calculate the corresponding *Precision at 1* (P@1), *Precision at 2* (P@2), *Mean*



(a) Object Function Value Update (b) Pairwise Error Rate (PER) Update (c) Query Weight α^Q Update

Figure 5: Scatter plot of CTR versus JRFL's prediction

Average Precision at 3 (MAP@3), *Mean Average Precision at 4* (MAP@4), and *Mean Reciprocal Rank* (MRR). Definitions of these metrics can be found in standard texts (e.g., [3]). In the editorial annotation data set, we treated the grade "Good" and above as relevant for precision-based metrics, and also used *discounted cumulative gain* (DCG) [12] as an evaluation metric. [4]

4.2 Analysis of JRFL

4.2.1 Convergency

We first demonstrate the convergency of the coordinate descent algorithm for JRFL model as described in Figure 4, which is the necessary condition for applying the proposed model in real ranking problems. We randomly divided the training preference pairs into two sets, one with 90k pairs for training and the rest 60k for testing. We fixed the trade-off parameter C in JRFL to be 5.0 (we also tried other settings for this parameter, smaller C would render us less iterations to converge, but the tendency of convergency is the same), relative convergency bound ϵ to be 10^{-5} and maximum iteration step S to be 50 in the coordinate descent algorithm. To study if the algorithm's convergence is sensitive to the initial state, we tried 3 starting points: 1) fixing the initial query weights $\alpha^{Q(0)}$ to be 1.0 (freshness only); 2) fixing $\alpha^{Q(0)}$ to be 0.0 (relevance only); 3) setting it uniformly between 0 and 1, by directly set all the query weights accordingly at step 0. We visualize the training process by illustrating the updating trace of object function defined in Eq(7), pairwise error rate on both training and testing set, and the mean/standard deviation of the updated query weights during the iterative optimization in Figure 5.

As demonstrated in Figure 5 that the proposed JRFL model converges during the coordinate descent optimization process, and such convergency does not depend on the initial state. From Figure 5(c), we can observe that the optimal query weight setting for this training set is usually around 0.4546 ± 0.0914. Hence, the random initialization converges fastest comparing with two other settings, since the initial state given by random is closest to this optimal setting.

In addition, it should be emphasized that, although the average of the converged value of α^Q is less than 0.5, it does *not* necessarily indicate freshness is less important than relevance in general for news search task, because the scales of the outputs of our freshness and relevance models may not be comparable. Therefore, only the order among the queries given by such learned query weights tell us their relative emphasis on relevance and freshness aspects.

Another phenomenon we observed in Figure 5(a) and (b) is that even though the object function decreased quickly after the first several iterations, the pairwise error rate needed more iterations to reach its optimal value. Furthermore, during these updates, there were some inconsistent updates between the object function value and pairwise error rate: while the object function value always decreased with more iterations, the pairwise error rate did not show the same monotonic behavior. This inconsistence is expected, because our object function in Eq (7) is a relaxed one: we do *not* directly minimize the pairwise error rate, which is computationally intractable, but we are trying to reduce the *prediction gap* between the mis-ordered pairs.

Table 4: Feature weights learned by JRFL

Feature	Type	Top 3 features
URL freshness	Neg	$\mathbf{age}_{\text{pubdate}}$(URL|Query) **LM@5**(URL|Query, t) **t-dist**(URL|Query)
Query model	Pos	**q_ratio**(Query|t) **pub_frq**(Query|t) **q_prob**(Query|t)
	Neg	**Ent**(Query|t) **pub_dev**(Query|d) **pub_mean**(Query|d)

Table 4 gives the top 3 positive and negative features from the newly proposed URL freshness features and query features, ordered by the learned weights. The weights in the linear model reflect the features' relative contribution to the final ranking decision. Because we only have 6 URL freshness features, and the learned weights for them are all negative, we only list the top 3 negative ones in this table.

The weights learned by the corresponding models are reasonable and consistent with our design: for URL freshness features, the smaller values of *Publication age* $\mathbf{age}_{\text{pubdate}}$(URL|Query), *Story coverage* **LM@5**(URL|Query, t) and *Relative age* **t-dist**(URL|Query) are, the more recent the news ar-

[4] According to Yahoo!'s business rule, the reported metrics are normalized accordingly; therefore only the relative improvement makes sense.

ticle is; and for the query features, the larger values of *query frequency* **q_prob**(Query|t) and *URL recency* **pub_frq**(Query|d), and the smaller values of *Distribution entropy* **Ent**(Query|t), *URL recency* **pub_mean**(Query|d) and **pub_dev**(Query|d) are, the more users and news reports start to focus on this event, and therefore the freshness quality becomes more important.

4.2.2 Relevance and Freshness Learning

Since our JRFL model does not rely on explicit relevance/freshness annotations, it is important to evaluate how well our relevance and freshness models can estimate each aspect separately. We separately used the relevance and freshness annotations on Aug. 9, 2011's query log as the test bed and utilized two GBRank models trained on Dong et al,'s relevance and freshness annotation data set accordingly (44,641 query-URL pairs) [8] as baseline methods. Because [8]'s data set does not contain the corresponding click information, those two GBrank models were trained without new query features. Our JRFL was trained on all the extracted click preference pairs.

Table 5: Performance on individual relevance and freshness estimation

	P@1	MAP@3	DCG@5
Relevance GBRank	0.9655	0.3422	14.6026
JRFL Relevance	0.8273	0.2291	14.7962
Freshness GBRank	0.9823	0.4998	18.8597
JRFL Freshness	0.9365	0.3106	19.8228

We observe mixed results in Table 5: the relevance and freshness modules inside JRFL have worse ranking performance than the purely relevance/freshness trained GBRank models at the top positions (lower P@1 and MAP@3), but similar cumulative performance, i.e., DCG@5 for both aspects. The reason for this result is that JRFL model has to account for the trade-off between relevance and freshness imposed by the queries during training, the most relevant or recent documents might not be treated as good training examples if their another aspects were not desirable. However, the purely relevance/freshness trained GBRank models do not have such constraints, and can derive the patterns to put the most relevant/recent documents at the top positions separately. As a result, those two GBRank models' performance can be interpreted as *upper bounds* for the individual ranking criteria (freshness/relevance) in this data set. When we reach the lower positions, those two types of ranking algorithms give the users quite similar utilities, i.e., under DCG@5 metric. Besides, we want to emphasize that such result is already very encouraging since the JRFL model successfully infers the relevance and freshness *solely from the clicks*, which confirms the soundness of our assumption in this work that users' click behavior is the joint consequence of examining the relevance and freshness of a news article for the given query; by properly modeling such relationships, we can estimate the relevance and freshness from the clickthroughs to certain extend.

4.2.3 Query Weight Analysis

There is no direct way for us to evaluate the correctness of the inferred query weights, since such information is not observable in the search log. Therefore, in this experiment, we investigate it in an indirect way. As we have discussed in the previous discussion, the order among the queries given by such weight reflects the query's relative emphasis over freshness aspect. Therefore, we ranked the queries in our training set according to the learned weights and list the top 10 (freshness-driven) and bottom 10 (relevance-driven) queries in Table 6.

Table 6: Query intention analysis by the inferred query weight

Freshness Driven	Relevance Driven
7-Jun-2011, china	5-Jul-2011, casey anthony trial summary
6-Jul-2011, casey anthony trial	9-Jul-2011, nascar qualifying results
24-Jun-2011, nba draft 2011	8-Jul-2011, burbank 100 years parade
28-Jun-2011, libya	10-Jul-2011 gas prices summer 2011
9-Jun-2011, iran	10-Jul-2011, bafta film awards 2011
6-Jun-2011, pakistan	2-Jul-2011, green lantern cast
13-Jun-2011, lebron james	9-Jul-2011, 2011 usga open leaderboard
29-Jun-2011, greece	3-Jul-2011, lake mead water level july 2011
27-May-2011, joplin missing	5-Jul-2011, caylee anthony autopsy report
6-Jun-2011, sarah palin	4-Jul-2011, aurora colorado fireworks 2011

at the first glance, it may be surprising to notice that most of the top ranked *freshness-driven* queries are the names of some countries and celebrities, e.g., "iran", "libya" and "lebron james", . But that is also quite reasonable: for those kind of queries, they are actually ambiguous, since there would be many candidate news reports related to different aspects of these queries. But when users issue such type of "ambiguous" queries in news search, they should be most interested in the recent updates about these countries and celebrities. We went back to check the most clicked URLs of these queries, and the clicks confirmed our assumption: the most clicked URLs for query "libya" were about the recent progress of libya war; and news articles covering the latest diplomatic affairs between U.S. and Iran got most clicks for the query "iran".

Table 7: Query length distribution under different query categories

Freshness Driven	Relevance Driven	ALL
1.446 ± 0.804	3.396 ± 1.309	2.563 ± 1.203

Another interesting finding from the learned query weights is that the length of *relevance-driven* queries is much longer than the *freshness-driven* queries. To validate this observation, we selected the top 500 *relevance-driven* and top 500 *freshness-driven* queries to calculate the corresponding query length distributions comparing to the length distribution of all the queries, and showed the results in Table

7. This result is consistent with our intuition: when users are seeking specific information, they tend to put more constraints (i.e., longer queries) to describe their information need; in contrast, when users are making recency search, they usually do not have a pre-determined mind about the events, so they often issue broad queries (i.e., shorter queries) about entities of their interest to see what is happening recently to those entities. Apparently, our query weight model is consistent with this intuition, and is able to distinguish these two typical searching scenarios. And this also reminds us query length is a good query feature to indicate the preference over freshness aspect.

4.3 Ranking Performance

To validate the effectiveness of the proposed JRFL model in real news search tasks, we quantitatively compare it with all our baseline methods on: random bucket clicks, normal clicks, and editorial judgments. All the click-based learning algorithms are trained on all 150K click preferences. Since all these models have several parameters to be tuned (e.g., the trade-off parameter C in both RankSVM and JRFL), we report their best performance on the corresponding testing set according to MAP@4 metric in the following results and perform t-test to validate the significance of improvement (against the runner-up performance accordingly).

First, we perform the comparison on the random bucket clicks, because such clicks are more trustable than normal clicks due to the removal of positional biases.

Table 8: Comparison On Random Bucket Clicks

Model	FreshDem	RankSVM	GBrank	JRFL
P@1	0.3413	0.3706	0.3882	**0.3969***
P@2	0.3140	0.3372	0.3477	**0.3614***
MAP@3	0.5301	0.5601	0.5751	**0.6012***
MAP@4	0.5859	0.6090	0.6218	**0.6584***
MRR	0.5899	0.6135	0.6261	**0.6335***

* indicates p-value<0.05.

From the results in Table 8, we can find the proposed JRFL model achieves encouraging improvement over the second best GBRank model, especially on MAP@4 the relative improvement is over 5.88%. This improvement confirms that properly integrating relevance and freshness can indeed improve the user's search satisfaction.

Table 9: Comparison On Normal Clicks

Model	FreshDem	RankSVM	GBrank	JRFL
P@1	0.3886	0.5981	0.5896	**0.6164***
P@2	0.2924	0.4166	0.4002	**0.4404***
MAP@3	0.4991	0.7208	0.6849	**0.7502***
MAP@4	0.5245	0.7383	0.7024	**0.7631***
MRR	0.5781	0.7553	0.7355	**0.7702***

* indicates p-value<0.05

Now we perform the same comparison on the normal click set, and we can observe similar improvement over other ranking methods, as shown in Table 9. Besides, from the results on both of these two click data sets, we can clearly observe that the click-preference-trained models generally outperform the freshness-demotion-trained GBRank model: JRFL's improvement on P@1 against the freshness-demotion-trained GBRank model is 16.2% and 58.6% on the random bucket click set and normal click set respectively. We have already analyzed the reason for this degenerated results in Figure 1 that a static grade demotion strategy can hardly guide the learning algorithm to achieve the optimal ranking results.

On the editorial annotation data set, to compare different model's performance, we mapped the separately annotated relevance and freshness labels into one single label by certain predefined strategy. In this experiment, we used the same freshness demotion strategy in Dong et al.'s work [8].

Table 10: Comparison On Editorial Annotations

Model	FreshDem	RankSVM	GBrank	JRFL
P@1	0.9184	0.9626	**0.9870**	0.9508
P@2	0.9043	0.9649	**0.9729**	0.9117
MAP@3	0.3055	0.3628	0.3731	**0.4137**
MAP@4	0.4049	0.4701	**0.4796**	0.4742
MRR	0.9433	0.9783	**0.9920**	0.9745
DCG@1	6.8975	7.9245	**8.1712***	7.2203
DCG@5	15.7175	17.2279	17.7468	**18.9397***

* indicates p-value<0.05.

From this result, we find that the freshness-demotion-trained GBRank model did not achieve the best performance on such "grade demoted" testing set either. This might be caused by the time gap between different annotations: [8]'s annotations were generated more than one year ago (February to May, 2009). The out-of-date annotation might not contain consistent relations between queries and document as in the new annotations. Besides, we also notice that the margin of improvement from JRFL becomes smaller comparing to the click-based evaluations. In the following, we perform some case studies to find out the reasons for the diminished improvement.

In Table 11, we illustrate one case of inferior ranking result for the query "afghanistan" from JRFL. We list the top 4 ranked results from JRFL together with the editorial relevance and freshness grades. (We have to truncate some of the URLs because they are too long to be displayed.)

Table 11: Case Study: Degenerated ranking results by JRFL for query "afghanistan"

URL	Relevance	Freshness
http://www.cbsnews.com/video/watch/?id=7376057nXXX	Good	Excellent
http://news.yahoo.com/afghanistan-helicopter-crash-why-army-used-chinook-half-180000528.html	Excellent	Excellent
http://news.yahoo.com/whatever-happened-civilian-surge-afghanistan-035607164.html	Excellent	Excellent
http://www.msnbc.msn.com/id/44055633/ns/world_news-south_and_central_asia/XXX	Perfect	Excellent

The freshness weight inferred by JRFL for this query is 0.7694, which is biased to freshness aspect. However, all those URLs' freshness grades are "Excellent", so that in the demoted final grades, the "ground-truth" ranking only depends on the relevance aspect. The predicted relevance score difference gets diminished by this biased freshness weight in JRFL: the JRFL predicted relevance score difference between the best document in "ground-truth" (last row in the table) versus JRFL ordering (first row in the table) is 0.44 while the corresponding freshness score difference is -0.31. As a result, JRFL gives a relatively "bad" ranking for this query.

In addition, we want to revisit the relationship between the predicted orders given by JRFL and CTR as we have done in Figure 2. This time, we draw the scatter plot between the JRFL predicted ranking scores and CTRs on the same set of URLs as in Figure 2.

The monotonic relationship between the predicted ranking and CTRs is much more evident than the one given by the demoted grades: URLs with lower CTRs concentrate more densely in the area with lower prediction scores, and the average Pearson correlation between the predicted ranking score and CTR across all the queries is 0.7163 with standard deviation 0.1673, comparing to the average of 0.5764 and standard deviation of 0.6401 in the the demoted grades.

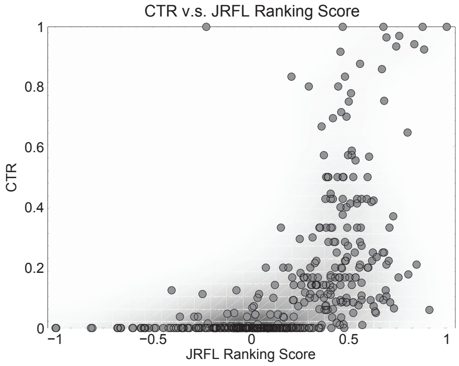


Figure 6: Scatter plot of CTR versus JRFL's prediction.

5. CONCLUSIONS

In this work, we proposed a joint learning framework, Joint Relevance Freshness Learning, for modeling the topical relevance and freshness, and the query-specific relative preference between these two aspects based on the clickthroughs for the news search task. Experiments on large-scale query logs and editorial annotations validate the effectiveness of the proposed learning method.

In this paper, we only instantiate the proposed joint learning framework by linear models, but many alternatives exist. It would be meaningful to employ other types of non-linear functions, for example, using Logistic functions can naturally avoid the range constrains over query weights in optimization.

Besides, in our current setting, the preference between relevance and freshness is assumed to be only query-dependent. It would be interesting to extend this to user-dependent, i.e., personalized search. By defining a proper set of user-related features, or profiles, the proposed JRFL can be easily applied in such user-centric retrieval environment. What is more, the proposed model can also be flexibly extended to other retrieval scenarios, where usefulness judgment is beyond pure topical relevance, such as opinions in blog search and distance in location search.

6. REFERENCES

[1] D. Agarwal, B.-C. Chen, P. Elango, and X. Wang. Click shaping to optimize multiple objectives. In *KDD*, 2011.

[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR*, 2006.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[5] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, pages 129–136. ACM, 2007.

[6] H. Chernoff and E. Lehmann. The use of maximum likelihood estimates in $\chi 2$ tests for goodness of fit. *The Annals of Mathematical Statistics*, pages 579–586, 1954.

[7] N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *SIGIR*, pages 95–104, 2011.

[8] 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 *WSDM*, pages 11–20, 2010.

[9] 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 *WWW*, 2010.

[10] M. Efron and G. Golovchinsky. Estimation methods for ranking recent information. In *SIGIR*, pages 495–504, 2011.

[11] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In *SIGIR*, 2004.

[12] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM TOIS*, 20(4):422–446, 2002.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[14] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*, pages 154–161, 2005.

[15] K. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36(6):779–808, 2000.

[16] N. Kanhabua and K. Nørvåg. Determining time of queries for re-ranking search results. *Research and Advanced Technology for Digital Libraries*, pages 261–272, 2010.

[17] A. Kulkarni, J. Teevan, K. Svore, and S. Dumais. Understanding temporal query dynamics. In *WSDM*, 2011.

[18] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM WSDM '11*, pages 297–306, 2011.

[19] X. Li and W. Croft. Time-based language models. In *CIKM*, pages 469–475, 2003.

[20] T. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[21] T. Moon, L. Li, W. Chu, C. Liao, Z. Zheng, and Y. Chang. Online learning for recency search ranking using real-time user feedback. In *CIKM*, pages 1501–1504, 2010.

[22] G. Salton and M. McGill. *Introduction to modern information retrieval.* McGraw-Hill, Inc., 1986.

[23] K. M. Svore, M. N. Volkovs, and C. J. Burges. Learning to rank with multiple objective functions. In *WWW*, 2011.

[24] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.

[25] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*, pages 287–294, 2007.