Session Based Click Features for Recency Ranking

Yoshiyuki Inagaki and **Narayanan Sadagopan** and **Georges Dupret** and **Ciya Liao**

Anlei Dong and **Yi Chang** and **Zhaohui Zheng**
Yahoo Labs
701 First Avenue
Sunnyvale, CA 94089
inagakiy,narayans,gdupret,ciyaliao,anlei,yichang,zhaohui@yahoo-inc.com

Abstract

Recency ranking refers to the ranking of web results by accounting for both relevance and freshness. This is particularly important for "recency sensitive" queries such as breaking news queries. In this study, we propose a set of novel click features to improve machine learned recency ranking. Rather than computing simple aggregate click through rates, we derive these features using the temporal click through data and query reformulation chains. One of the features that we use is *click buzz* that captures the spiking interest of a url for a query. We also propose time weighted click through rates which treat recent observations as being exponentially more important. The promotion of fresh content is typically determined by the query intent which can change dynamically over time. Quite often users query reformulations convey clues about the query's intent. Hence we enrich our click features by following query reformulations which typically benefit the first query in the chain of reformulations. Our experiments show these novel features can improve the NDCG5 of a major online search engine's ranking for "recency sensitive" queries by up to 1.57%. This is one of the very few studies that exploits temporal click through data and query reformulations for recency ranking.

Introduction

Web search engines are expected to return relevant search results for a query. A user who issues a query can have varying information needs depending on the context. For example, a user querying for "circus" could be interested in local circus shows in an area or may be looking for information about Britney Spears' album "circus". Such queries that have temporally varying intent are called "recency sensitive" queries. Ranking of search results for these queries is extremely challenging. For example in the case of query "circus", users might prefer to see a result about Britney Spears' album closer to the day the album was released. However if the album was released in the distant past the user may prefer results about the local circus shows in his geographical area. The problem of ranking documents by relevance while taking freshness into account is called recency ranking (Dong et al. 2009). Recency ranking on web-scale offers several unique and challenging research problems. First, ranking algorithms for recency sensitive queries need to satisfy both relevance and freshness requirements. In order to ensure robustness, the system needs to promote recent content only when appropriate, so as to avoid degrading relevance for other queries classes.

Quite often a machine learned approach is used for general web search ranking (Burges et al. 2005) (Zheng et al. 2007). This approach usually employs several features to predict a relevance score. The number of features can range in the order of a few thousands and can be obtained from link analysis, semantic matching of query to document, etc. Several studies have also demonstrated the utility of user click feedback in improving the relevance of machine learned ranking (Joachims 2002a) (Agichtein, Brill, and Dumais 2006). Since most of these studies focus on improving ranking for general web search, they compute aggregate click through rate features. In this work we focus on the design of several novel click features that can be used to improve machine learned recency ranking. Due to the temporal aspect of recency ranking these features are derived from the time series of click data. One of the features is *click buzz* that captures the spiking interest of a url for a query. We also propose a simple exponential weighted scheme for computing click through rates that over emphasizes recent data.

Typically users issue one or more queries to satisfy their information needs (Jones and Klinkner 2008). These query sequences (or chains) contain a wealth of information that can be further exploited for computing click features. For example, in the case of the query "circus", if the user is not satisfied with the current search results, he/she might reformulate the query to "britney spears circus" to get the desired search result. In this case it may be desirable to *smooth* the clicks for "britney spears circus" and attribute it to the original query "circus". Intuitively this mechanism which we call *"smoothing"* would help in improving the ranking of queries that have temporally varying intents. Our experiments show that the proposed set of click features along with *smoothing* across query chains does indeed produce substantial relevance gains (up to 1.57%) over a major online search engine's recency ranking.

Related Work

Recency Ranking: There has been a lot of studies on improving general Web search ranking. State of the art work in this area employs machine learning to automatically construct a ranking function which optimizes retrieval performance metrics (Burges et al. 2005; Joachims 2002a; Zheng et al. 2007; Y. Cao and Hon 2006). Machine-learned rankers have demonstrated significant and consistent gains over many manually tuned rankers for a variety of data sets. Optimization usually is formulated as learning a ranking function from human labeled preference data in order to minimize a loss function (e.g., the number of incorrectly ordered documents pairs in the training data). Different algorithm cast the preference learning problem from different points of view. For example, RankSVM (Joachims 2002b) used support vector machines; RankBoost (Freund et al. 1998) applied the idea of boosting from weak learners; GBrank (Zheng et al. 2007) used gradient boosting with decision trees; RankNet (Burges et al. 2005) used gradient boosting with a neural network. Improving recency ranking has also attracted a lot of interest. There are two main aspects to this ranking problem: detecting if the query is recency sensitive (i.e. the user prefers more recent content) and appropriately ranking recent content along with other results. Diaz proposed a solution to integrate search results from a news vertical search into web search results, where the news intent is detected by either inspecting query temporal dynamics or using click feedback (Diaz 2009). The main focus of their work is to detect if a query is "newsworthy". Zhang *et al.* proposed a ranking score adjustment method on year-qualified-queries, in which a few simple but effective adjustment rules are applied to the ranking results based on the timestamps extracted from the documents(Zhang et al. 2009). Dong *et al.* studied the impact of different machine learning strategies and algorithms on recency ranking (Dong et al. 2009). In this work we propose a set of novel temporal click features that can be used to improve the machine learned ranking for all time-sensitive queries.

Query Chain and Query Segmentation: Segmenting the query stream of a user into sets of related queries has many applications, such as query recommendation, user profiling and personalization. He and Göker (He and Göker 2000) studied various timeouts to segment user sessions, and later extended their work (He, Göker, and Harper 2002) to take features like the overlap between terms in two consecutive queries into consideration. Radlinski and Joachims (Radlinski and Joachims 2005) observed that users often issue a series of queries with a similar information need. They referred to this sequence of related, often reformulated queries as *query chain*. They presented a simple method to automatically detect query chains in query and click through logs and showed how clicks inferred query chains could be used to train better ranking functions. Jones and Klinkner (Jones and Klinkner 2008) proposed an automated method for segmenting sessions based on a machine learning approach. More recently, Boldi et al. (Boldi et al. 2008) proposed a novel query segmentation method by building *query-flow graph* and using a machine learning approach to identify query chains. Our intuition is that these user reformulation chains can help in identifying temporal changes in the query intent. In this work we leverage query chains to *smooth* our proposed click features.

Recency Data

This section briefly describes the recency training data, including evaluation guideline, recency features, recency data collection.

Collecting training data collection for recency ranking is different from regular web search ranking. For regular web search ranking, the relevance judgement for a query-url pair is usually static over time because document freshness does not affect the user satisfaction. The judgement for a recency-sensitive query-url pair, however, should incorporate the freshness of the document. Thus for recency ranking, we need editorial labels for tuples of the form $\langle \text{query}, \text{url}, t_{\text{query}} \rangle$ instead of only $\langle \text{query}, \text{url} \rangle$, where t_{query} is the time at which the query is issued. If the judging time t_j of $\langle \text{query}, \text{url} \rangle$ is far behind query issue time t_{query}, it is impractical for the editors to make a reliable judgement. Therefore, we collect sets of recency data periodically instead of collecting them all at once. Each time we collect a set of $\langle \text{query}, \text{url}, t_{\text{query}} \rangle$ tuples, we ask editors to judge the query-url pairs immediately, so that the query issue time and judging time are as close as possible. Collecting recency data periodically can also prevent the data distribution from being too biased towards a short period of time. For example, within one day, there are many similar queries related to the same breaking news, which are very different from the query distribution over a longer time span.

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

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

The advantages of this recency-demotion grading method include:

- Recency is incorporated into the modeling of the the ranking problem
- Recency can also be decoupled from overall relevance so that recency and non-recency relevance can be analyzed separately during learning.

To best represent page recency with available information, Dong *et al.* used four categories of recency-related features besides traditional relevance ranking features: *timestamp features*, *linktime features*, *webbuzz features* and *page classification features*. For timestamp features and linktime features, time values were extracted from web page content and page discovery log respectively; these two categories of features were used as an attempt to represent page freshness. Webbuzz features are similar to page and link *activity* used in the T-Rank algorithm and represent the update rates of pages and links, reflecting the recent popularity of the pages (Berberich, Vazirgiannis, and Weikum 2005). They also used page classification features that are closely related to page recency, and are useful in ranking model learning to help us appropriately use recency features for different classes of pages. They did not use any information from the user query log data. In this study we build upon their work by proposing a set of novel features derived from user click logs that can be used to improve recency ranking.

Click Features for Recency Ranking

In this study we demonstrate the utility of user clicks for recency ranking. Since modeling time is essential for recency ranking, we design click features that are derived from the time series of click data. We also use query reformulation *smoothing* to exploit user feedback for these time-sensitive queries.

The most common click feature used in several studies is click through rate that is defined as follows:

Definition 1 *Click Through Rate* CTR*(query, url) is defined as the ratio of the number of sessions with clicks on the given url for the given query and the total number of sessions in which the url is viewed for the query* [1].

More formally, CTR(query,url) is computed as:

$$\texttt{CTR} = \frac{\sum_{i=1}^{T} c_i}{\sum_{i=1}^{T} v_i}$$

where T is period of data collection, c_i and v_i are the number of clicks and views respectively for (query,url) on day i.

Apart from CTR, in this study, we also propose new click features such as Only Click Through Rate and Click Attractivity which are defined below:

Definition 2 *Only Click Through Rate* $\texttt{CTR}_o$*(query,url) is the ratio of the number of sessions in which the user clicked only on the given url (and nothing else) for the given query and the total number of sessions in which the url is viewed for the query.*

Definition 3 *Click Attractivity* ATTR*(query,url) is the ratio of the number of sessions with clicks on the given url for the given query and the total number of sessions in which the url was either clicked or examined* [2]

Along the same lines, we define similar features for the query, host pair. These features are called CTRH, $\texttt{CTRH}_o$ and ATTRH respectively.

Specifically for recency ranking, we propose a time varying CTR with a simple weighting strategy. Click and view counts for a day are exponentially weighted to emphasize recent observations. The exact form of weighting we use is the following:

$$\texttt{CTR}^w(\text{query, url, } t_{query}) = \frac{\sum_{i=1,v_i>0}^{t_{query}}(c_i \cdot (1+x)^{i-t_{query}})}{\sum_{i=1,v_i>0}^{t_{query}}(v_i \cdot (1+x)^{i-t_{query}})}$$

where $x > 0$.

If $x = 0$, $\texttt{CTR}^w = \texttt{CTR}$, the traditional unweighed CTR. We can vary x based on how steep we want to overemphasize recent observations.

In our experiments we use weighted versions of all earlier mentioned CTRs such as $\texttt{CTR}^w$, $\texttt{CTR}_o^w$, $\texttt{ATTR}^w$, $\texttt{CTRH}^w$, $\texttt{CTRH}_o^w$ and $\texttt{ATTRH}^w$.

We also define the *click buzz* feature to identify whether a page, a site or a host that receives an unusual level of attention from the users compared to the past. Measure of attention can be the number of clicks on the document, number of clicks on the query document pair, on the host, etc. Take the particular example of clicks on a query url pair c_t, the number of clicks during the time interval indexed by t. We first compute the average and the variance of the statistic of interest over T periods of time:

$$\begin{aligned} \bar{c}_T &= \frac{1}{T}\sum_{t=1}^{T} c_t \\ \texttt{VAR(c)} = \sigma_T^2(c) &= \frac{1}{T}\sum_{t=1}^{T}(c_t - \bar{c}_T)^2 \end{aligned}$$

The *buzz* of a feature is defined as the deviation from the empirical average of the feature value over a period of time, normalized by the standard deviation over the same period. It is therefore a measure of how atypical the feature value is on a given day, compared with its value on average.

The buzz feature is computed as follows

$$b(\text{query, url, } t_{query}) = \frac{c_{t_{query}} - \bar{c}_{t_{query}}}{\sigma_{t_{query}}(c)}$$

Naturally, we can compute the *buzz* of any feature varying over time, for example the number of host clicks, number of times a query is issued, etc. (see Figure 1).

It is important to note that the proposed click features for a given (query, url, t_{query}) triplet is computed using only the user click data till t_{query}. These click features favor urls that have been of recent interest for a query. One of the challenges in recency ranking is to model changes in the query intent. We believe that looking at query reformulations can be a valuable source of information for this purpose. We discuss them in the next section.

Click Smoothing using Query Chains

In most analysis of web search relevance, a single query is the unit of user interaction with search engines. However, users tend to issue multiple queries in order to complete their task (Jones and Klinkner 2008). Jones *et al.* defined 2 classes of query chains, namely *goals* and *missions* (Jones and Klinkner 2008). A goal is an atomic information need that results in one or more queries while a mission is a related set of information needs that results in one or more

[1]Every time a url is displayed on the first page, it is counted as a view

[2]A url is considered examined if its position is above the maximum click position in a session.

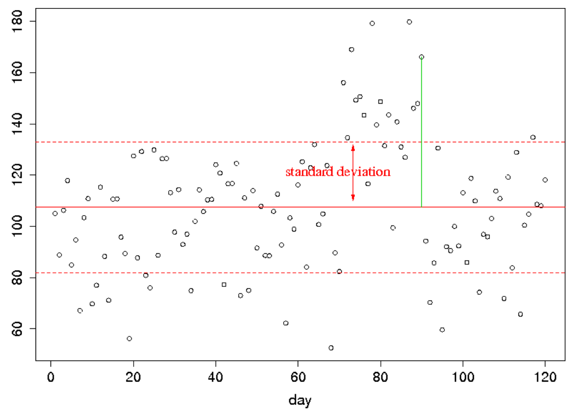

Figure 1: The *buzz* of CTR for a given query. The dotted lines represent one standard deviation around the mean. The right most vertical segment represents the deviation of a particular observation from the average. The largest the deviation counted in standard deviation units, the most unusual the click behavior.

Table 1: URLs that had been clicked for the query, "Circus".

```
u1=www.ringling.com
u2=www.youtube.com/watch?v=1zeR3NSYcHk
u3=en.wikipedia.org/wiki/Circus_(Britney_Spears_album)
u4=www.metrolyrics.com/circus-lyrics-britney-spears.html
u5=www.youtube.com/
u6=en.wikipedia.org/wiki/Circus
u7=britneyspearscircus.net/
```

goals. For instance, the mission of a user may be to organize a trip to Hawaii. Any one user mission can (but does not need to) be made up of several goals, and goals in the mission may be to find a hotel in Hilo, look at Kona restaurant reviews and check the NOAA Hawaii weather site.

Once we successfully identify task boundaries and segment a long sequence of queries into query chains, all queries in a query chain are presumably issued to satisfy a single information need by the user. Therefore, clicks made on any URLs in search results for any query in the query chain can be used to enrich the clickthrough data for the first query in the chain.

Consider the following example: On November 28th, 2008, Britney Spears released her new album called "Circus". Our query logs shows that the query "Circus" got popular a few days later (see Figure 2).

About the same time, URLs about Britney Spears' "Circus" started to be viewed and clicked more (see Figure 3 and 4). Since users could not get URLs about her album with the query "Circus", they issued a series of queries by refining the previous query until they got URLs about the album and clicked those URLs.

This series of queries to retrieve URLs about the album constitute one query chain. In this case, the goal of the user is to obtain results for Britney Spears' circus album. We can take the URLs that were clicked for "Circus Album" or

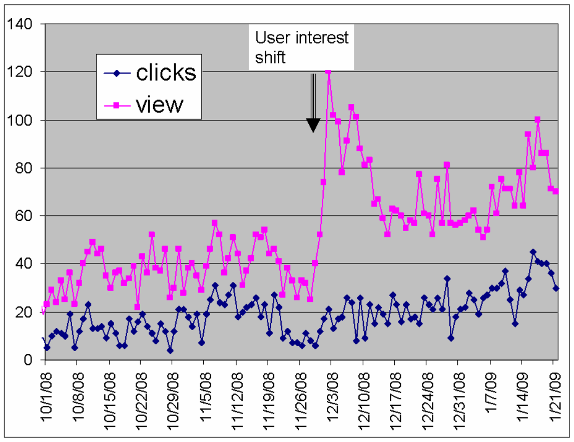

Figure 2: Users got interested in "Circus" a few days after the release of "Circus"album.

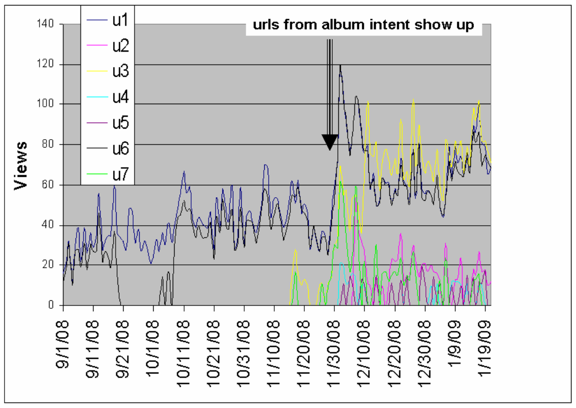

Figure 3: URLs (u2, u3, u4 and u7 in Table 1) about "Circus" album get more views.

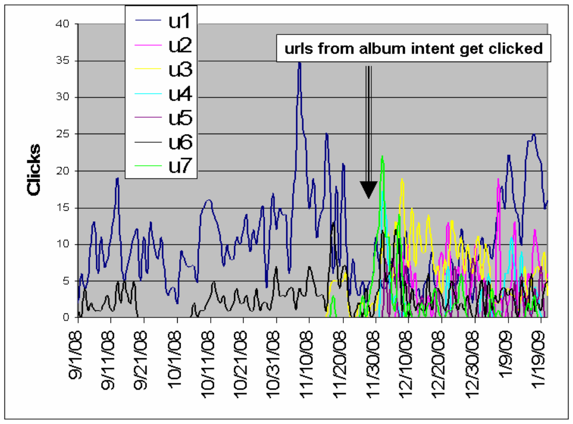

Figure 4: URLs (u2, u3, u4 and u7 in Table 1) about "Circus" album get more clicks.

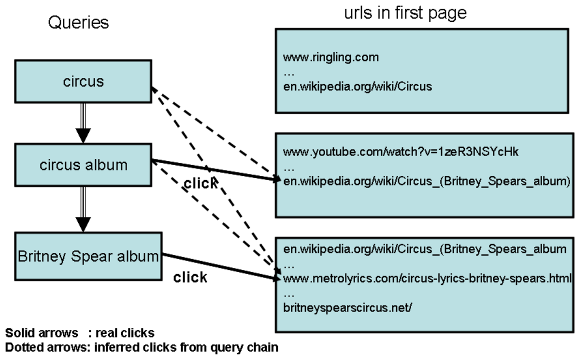
Queries
urls in first page
circus
circus album
Britney Spear album
click
click
www.ringling.com
...
en.wikipedia.org/wiki/Circus
www.youtube.com/watch?v=1zeR3NSYcHk
...
en.wikipedia.org/wiki/Circus_(Britney_Spears_album)
en.wikipedia.org/wiki/Circus_(Britney_Spears_album
...
www.metrolyrics.com/circus-lyrics-britney-spears.html
...
britneyspearscircus.net/
Solid arrows : real clicks
Dotted arrows: inferred clicks from query chain

Figure 5: Smoothing Clicks for "Circus Album" or "Britney Spears Album" for "Circus".

"Britney Spears Album" in the same query chain as those clicked for "Circus" (see Figure 5). We call this process of attributing clicks from one query in the query chain to another query (preferably the first one) as *smoothing*.

We evaluate 3 different *smoothing* mechanisms: the simplest 30 minute timeouts and the more complex ones, *goals* and *missions* based on (Jones and Klinkner 2008) and (Boldi et al. 2008).

Experiments

The training and testing data sets for recency consists of query-URL pairs that were judged in a *time-sensitive* manner. The training data set consists of 53,775 query-url pairs and 3291 queries that were made unique with their judged dates. The testing data set consists of 29,460 query-url pairs and 1771 queries that were made unique with judged dates.

We use the GBrank algorithm for the ranking model learning, as GBrank is one of the most effective learning-to-rank algorithms (Zheng et al. 2007). For the purpose of better readability, we briefly introduce the basic idea of GBrank. For each preference pair $< x, y >$ in the available preference set $S = \{< x_i, y_i > | x_i \succ y_i, i = 1, 2, ..., N\}$, x should be ranked higher than y. In GBrank algorithm, the problem of learning ranking functions is to compute a ranking function h , so that h matches the set of preference, i.e, $h(x_i) \geq h(y_i)$, if $x \succ y$, $i = 1, 2, ..., N$ as many as possible. The following loss function is used to measure the risk of a given ranking function h

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

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

We use *normalized discounted cumulative gain* (NDCG) which is a variant of the popular *discounted cumulative gain* (DCG) metric to evaluate the quality of the ranking model (Jarvelin and Kekalainen 2002). NDCG is defined as

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

where i is the position in the document list, G_i is the function of relevance grade. Z_n is a normalization factor, which is used to make the NDCG of the ideal ranked list to be 1. We use $\mathbf{NDCG}_5$ to evaluate the ranking results.

We used the state of the art recency ranking model developed by Dong *et al.* (Dong et al. 2009) as the baseline ranking function for comparison, but we did not use click buzz features. We evaluated our proposed set of features across several combinations of x (for exponential weighting) and click *smoothing* mechanisms. We first fix a *smoothing* mechanism and generate all features using the same mechanism.

Our first set of experiments were designed to evaluate the impact of exponential weighting of the click features. Hence we augmented the baseline ranking function with click features that did not have exponential weighting (i.e. $x = 0$) and did not use any *smoothing* or *Click buzz* features were excluded from these experiments. In these experiments, we varied the value of x in the range of $0 - 2$ in steps of 0.2. We also generated our click features by smoothing clicks using different query segmentations such as timeouts, missions and goals using the research of (Jones and Klinkner 2008) (Boldi et al. 2008). Table 2 shows that non zero values of x does indeed help recency ranking. In some cases, the gain in NDCG5 can be as high as 1.08% ($x = 0.4$ and using mission click *smoothing*). Our evaluation shows that exponential weighting helps in improving NDCG5 for recency ranking across all the mechanisms for *smoothing* clicks considered in this study. In particular goal based *smoothing* almost always provides statistically significant gains over the baseline function.

Table 2: Effect of x on percentage improvement in NDCG5. All models were trained without click buzz features. The baseline is $x = 0$, i.e., with CTR features that are not time weighted or *smoothed*. The numbers in parenthesis are the p-values. Bold entries are significant beyond 0.05 level of significance.

x	Timeouts	Missions	Goals
0.2	**0.67 (0.019)**	**0.92 (0.009)**	**0.71 (0.043)**
0.4	**0.74 (0.026)**	**1.08 (0.001)**	**0.80 (0.018)**
0.6	**0.67 (0.027)**	**0.79 (0.021)**	**0.90 (0.010)**
0.8	**0.95 (0.005)**	**0.95 (0)**	**1.03 (0.003)**
1.0	0.65 (0.117)	**0.75 (0.027)**	**1.00 (0.003)**
1.2	0.55 (0.073)	0.62 (0.059)	**0.85 (0.010)**
1.4	0.59 (0.062)	0.59 (0.119)	**0.93 (0.002)**
1.6	**0.75 (0.021)**	**0.84 (0.042)**	**0.81 (0.029)**
1.8	**0.63 (0.039)**	**0.67 (0.047)**	**0.96 (0.003)**
2.0	**0.65 (0.045)**	**0.64 (0.022)**	**1.08 (0.002)**
3.0	**0.71 (0.036)**	**0.89 (0.019)**	**0.86 (0.004)**
4.0	**0.76 (0.039)**	0.78 (0.051)	0.64 (0.084)

We also evaluated the effect of the *click buzz* features across different values of x (for exponential weighting) and different *smoothing* strategies. Table 3 shows that the *click buzz* features provide additional improvements (with better statistical significance) over the same baseline function. The best gains in NDCG5 (1.57%) are observed for $x = 0.8$ and goal based click *smoothing*.

Table 3: Effect of x on percentage improvement in NDCG5 together with click buzz features. Baseline is the same as used in Table 2.

x	Timeouts	Missions	Goals
0.2	0.56 (0.178)	**0.99 (0.005)**	**1.13 (0.001)**
0.4	**0.72 (0.021)**	**1.02 (0.002)**	**0.90 (0.008)**
0.6	**0.87 (0.007)**	**1.05 (0.003)**	**1.12 (0)**
0.8	**0.83 (0.003)**	**1.23 (0)**	**1.57 (0)**
1.0	**0.67 (0.016)**	**1.18 (0)**	**1.08 (0.003)**
1.2	0.63 (0.053)	**0.85 (0.014)**	**1.00 (0.002)**
1.4	**0.85 (0.003)**	**1.07 (0.001)**	**0.98 (0.001)**
1.6	**0.86 (0.006)**	**1.30 (0.0002)**	**0.92 (0.019)**
1.8	**0.75 (0.014)**	**1.03 (0.006)**	**0.94 (0.001)**
2.0	**0.70 (0.027)**	**1.13 (0)**	**0.90 (0.028)**
3.0	**1.04 (0)**	**0.97 (0.004)**	**1.00 (0.010)**
4.0	**0.91 (0.003)**	**1.06 (0.002)**	**1.12 (0.002)**

Discussions and Conclusions

In this study we propose novel temporal click features [3] for recency ranking. Rather than computing simple aggregate click through rates, features are derived using the temporal click through data and query reformulations. One of the features that we use is *click buzz* that captures the spiking interest of a url for a query. We also propose time weighted click through rate which treats recent observations as being exponentially more important. We enrich our click features by following query reformulations which typically benefit the first query in the chain of reformulations. Our experiments demonstrate that these features can improve NDCG5 of a major online search engine's recency ranking by up to 1.57%.

As part of future work, we would like to evaluate several other time series properties such as auto-correlation as features for recency ranking. Another interesting problem would be to automatically learn the exponential weighting scheme based on query or query categories. It would be interesting to evaluate these ideas for general web search ranking as well.

References

Agichtein, E.; Brill, E.; and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. *Proc. of ACM SIGIR Conference*.

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

Boldi, P.; Bonchi, F.; Castillo, C.; Donato, D.; Gionis, A.; and Vigna, S. 2008. The query-flow graph: model and applications. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*.

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

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

Dong, A.; Chang, Y.; Zheng, Z.; and Gilad Mishne, Jing Bai Karolina Buchner, R. Z. C. L. G. L. 2009. Towards recency ranking in web search. *WSDM*.

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

He, D., and Göker, A. 2000. Detecting session boundaries from web user logs. In *Proceedings of the BCS-IRSG 22nd annual colloquium on information retrieval research*.

He, D.; Göker, A.; and Harper, D. 2002. Combining evidence for automatic web session identification. *Inf. Process. Manage.* 38(5):727–742.

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

Joachims, T. 2002a. Optimizing search engines using clickthrough data. *Proc. of ACM SIGKDD Conference*.

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

Jones, R., and Klinkner, K. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*.

Radlinski, F., and Joachims, T. 2005. Query chains: Learning to rank from implicit feedback. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*.

Y. Cao, J. Xu, T.-Y. L. H. L. Y. H., and Hon, H.-W. 2006. Adapting ranking svm to document retrieval. *Proceedings of ACM SIGIR conference*.

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

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

[3] Because the proposed click features are all time-sensitive, how fast and often they can be generated seems to be an issue. However, a large-scaled commercial search engine can update the click-based features every hour. Therefore, they can reflect the latest popularity of the documents.