Beyond Clicks: Dwell Time for Personalization

Xing Yi Liangjie Hong Erheng Zhong Nathan Nan Liu Suju Rajan
{xingyi, liangjie, erheng, nanliu, suju}@yahoo-inc.com
Personalization Sciences, Yahoo Labs, Sunnyvale, CA 94089, USA

ABSTRACT

Many internet companies, such as Yahoo, Facebook, Google and Twitter, rely on content recommendation systems to deliver the most relevant content items to individual users through personalization. Delivering such personalized user experiences is believed to increase the long term engagement of users. While there has been a lot of progress in designing effective personalized recommender systems, by exploiting user interests and historical interaction data through implicit (item click) or explicit (item rating) feedback, directly optimizing for users' satisfaction with the system remains challenging. In this paper, we explore the idea of using item-level dwell time as a proxy to quantify how likely a content item is relevant to a particular user. We describe a novel method to compute accurate dwell time based on client-side and server-side logging and demonstrate how to normalize dwell time across different devices and contexts. In addition, we describe our experiments in incorporating dwell time into state-of-the-art learning to rank techniques and collaborative filtering models that obtain competitive performances in both offline and online settings.

Categories and Subject Descriptors: H.3.5 [Information Storage and Retrieval]: Online Information Services

General Terms: Theory, Experimentation

Keywords: Content Recommendation, Personalization, Dwell Time, Learning to Rank, Collaborative Filtering

1. INTRODUCTION

Content recommendation systems play a central role in today's Web ecosystems. Companies like Yahoo, Google, Facebook and Twitter are striving to deliver the most relevant content items to individual users. For example, visitors to these sites are presented with a stream of articles, slideshows and videos that they may be interested in viewing. With a personalized recommendation system, these companies aim to better predict and rank content of interest to users by using historical user interactions on the respective sites. The underlying belief is that personalization increases long term user engagement and as a side-benefit, also drives up other aspects of the services, for instance, revenue. Therefore, there has been a lot of work in designing and improving personalized recommender systems.

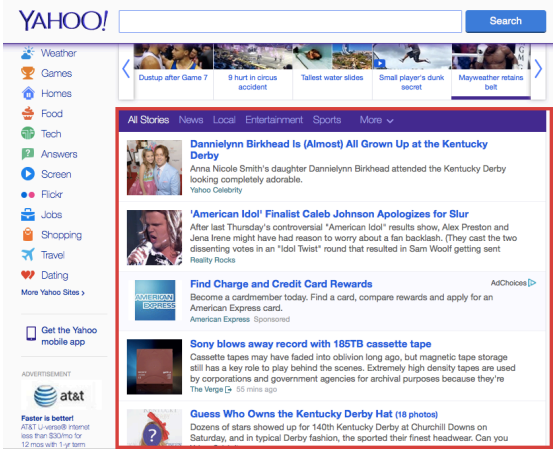

Figure 1: A snapshot of Yahoo's homepage in U.S. where the content stream is highlighted in red.

Traditionally, simplistic user feedback signals, such as click through rate (CTR) on items or user-item ratings, have been used to quantify users' interest and satisfaction. Based on these readily available signals, most content recommendation systems essentially optimize for CTR or attempt to fill in a sparse user-item rating matrix with missing ratings. Specifically for the latter case, with the success of the Netflix Prize competition, matrix-completion based methods have dominated the field of recommender systems. However, in many content recommendation tasks users rarely provide explicit ratings or direct feedback (such as 'like' or 'dislike') when consuming frequently updated online content. Thus, explicit user ratings are too sparse to be usable as input for matrix factorization approaches. On the other hand, item CTR as implicit user interest signal does not capture any post-click user engagement. For example, users may have clicked on an item by mistake or because of link bait, but are truly not engaged with the content being presented. Thus, it is arguable that leveraging the noisy click-based user engagement signal for recommendation can achieve the best long term user experience. In fact, a recommender system needs to have different strategies to optimize short term metrics like CTR and long term metrics like how many visits a user would pay in several months. Thus, it becomes critical to identify signals and metrics that truly capture user satisfaction and optimize these accordingly.

We argue that the amount of time that users spend on content items, the dwell time, is an important metric to measure user en-

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645724.

gagement on content and should be used as a proxy to user satisfaction for recommended content, complementing and/or replacing click based signals. However, utilizing dwell time in a personalized recommender system introduces a number of new research and engineering challenges. For instance, a fundamental question would be how to measure dwell time effectively. Furthermore, different users exhibit different content consumption behaviors even for the same piece of content on the same device. In addition, for the same user, depending on the nature of the content item and the context, the user's content consumption behavior can be significantly different. Therefore, it would be beneficial to normalize dwell time across different devices and contexts. Also, recommender systems usually employ machine learning-to-rank (MLR) techniques and collaborative filtering (CF) models, with a wide range of features, to obtain state-of-the-art performance. Using dwell time in these frameworks is not straight-forward.

In this paper, we use the problem of recommending items for the content feed or stream on the Yahoo's homepage, shown in Figure 1, as a running example to demonstrate how dwell time can be embedded into a personalized recommendation system. We have designed several approaches for accurately computing item-level user content consumption time from large-scale web browsing log data. We first use the log data to determine when each item page gains or loses the user attention. Capturing the user's attention on an item page enables us to compute per-user item-level dwell time. In addition, we leverage content consumption dwell time distributions of different content types for normalizing users' engagement signals, so that we can use this engagement signal for recommending multiple content type items to the user in the content stream. We then incorporate dwell time into machine learning-to-rank (MLR) techniques and collaborative filtering (CF) models. For MLR, we propose to use per-user item-level dwell time as the learning target, which can be easily considered in all existing MLR models, and demonstrate that it can result in better performances. For CF, we use dwell time as a form of implicit feedback from users and demonstrate that a state-of-the-art matrix factorization model to incorporate this information can yield competitive and even better performances than the click-optimized counterpart. To be more specific, we have made the following contributions in this paper:

- A novel method to compute fine-grained item-level user content consumption time for better understanding users' interests is proposed.
- A novel solution to normalize dwell time for multiple content-type items across different devices is proposed and presented.
- An empirical study of dwell time in the context of content recommendation system is presented in this paper.
- A MLR framework to utilize dwell time is proposed and its effectiveness in real-world settings is demonstrated.
- A CF framework to utilize dwell time is proposed and its effectiveness against non-trivial baselines is presented.

The paper is organized as follows. In §2, we review three related research directions. In §3, we demonstrate how dwell time can be measured and present some of its interesting characteristics. In the following sections, we show two important use cases for dwell time. In §4, we show how dwell time can be used in MLR to obtain superior performance than the models that optimize CTR. In §5, we plug dwell time into the state-of-the-art CF models and demonstrate that we can obtain competitive performance. We conclude the paper in §6.

2. RELATED WORK

In this section, we review three related research directions. First, we examine how dwell time is studied and used in web search or IR domains. We will carefully analyze which of the existing practices and experiences, on dwell time computation, can be utilized in the context of personalization. We then list several pointers for MLR as it has been extensively studied in the past decade, followed by a brief discussion on CF, paying special attention to how implicit user feedback is used in CF.

Dwell Time in Other Domains: A significant amount of previous research on web search has investigated using post-click dwell time of each search result as an indicator of its relevance for web queries and how it can be applied for different web search tasks. All such previous research focused on examining the dwell time's utility for improving search results. For instance, White and Kelly [15] demonstrated that using dwell time can potentially help improve the performance of implicit relevance feedback. Kim et al. [8] and Xu et al. [17, 18] showed that using webpage-level dwell time can help personalize the ranking of the web search results. Liu et al. [11] investigated the necessity of using different thresholds of dwell time, in order to derive meaningful document relevance information from the dwell time for helping different web search tasks. To the best of our knowledge, we are the first to use dwell time for personalized content recommendation. Furthermore, we consider different types of content (news articles, slideshows and videos), present several approaches to accurately measure content consumption time, and use the dwell time for understanding users' daily habit and interests. Most recently, Youtube has started to use users' video time spent instead of the click event [1] to better measure the users' engagement with video content. In contrast, we focus on

Learning To Rank in Web Search: The field of MLR has significantly matured in the past decade, mainly due to the popularity of search engines. Liu [12] and Li [10] provide an in-depth survey on this topic. Here, we point out that a fundamental issue with all existing MLR models is that they all optimize for *relevance*, an abstract yet important concept in IR. In the standard setting, the "relevance" between a particular query and a list of documents is objective and the same for all users. For IR, "relevance" is judged by human experts through a manual process and is difficult to scale to millions of real queries. In order to personalize IR, a natural alternative to "relevance" is to optimize CTR. In this paper, we explore the possibility of optimizing for dwell time under the existing framework of gradient boosted decision trees [6]. However, other MLR models can also be used such as pair-wise models (e.g., RankBoost [5] and AdaRank [16]) and list-wise models (e.g., RankNet [3] and ListNet [4]). Note that, we do not seek to propose new MLR models, but instead show the advantage of utilizing dwell time in existing models.

Collaborative Filtering: In CF systems, users' satisfaction with the items is usually not considered. Almost all previous work in CF (e.g., [9, 1]) take only explicit feedback such as ratings, or "implicit" click-based feedback into account. Hu et al. [7] considered implicit feedback signal, such as whether a user clicks or reviews an item, and incorporated it into the matrix factorization framework. Rendle et al. [13] proposed a learning algorithm for binary implicit feedback datasets, which is essentially similar to AUC optimization. None of them went beyond binary implicit feedback to investigate the interactions between users and items. The approach of Yin et al. [19] is the closest to our work. In that paper, the authors used a graphical model on the explicit feedback signals and dwell time

[1] http://youtubecreator.blogspot.com/2012/08/youtube-now-why-we-focus-on-watch-time.html

Table 1: Client-side Logging Example

User Behaviors	Client-side Events
A user opens a news article page.	$\{\texttt{DOM-ready}, t_1\}$
He reads the article for several seconds.	$\{\texttt{Focus}, t_2\}$
He switches to another browser tab or a window to read other articles.	$\{\texttt{Blur}, t_3\}$
He goes back to the article page and comments on it.	$\{\texttt{Focus}, t_4\}$
He closes the article page, or clicks the back button to go to another page.	$\{\texttt{BeforeUnload}, t_5\}$

data to predict the user's score. Our work is different in that our model does not require the presence of explicit user feedback.

3. MEASURING ITEM DWELL TIME

In this section, we describe how dwell time can be measured from web logs and show its basic characteristics.

3.1 Dwell Time Computation

Accurately computing item-level dwell time from web-scale user browsing activity data is a challenging problem. As an example, most modern browsers have a multi-tabbed interface in which users can open multiple stories simultaneously and switch between them. In the multi-tabbed setting, figuring out the tab that captured the user's attention is non-trivial. In this paper we describe two complementary methods to derive dwell time, one via client-side logging and the other via server-side logging. We have also conducted a simple study comparing these two approaches. Although client-side logging can capture fine-grained user behavior and has the potential of being highly useful, there is a lot of dependency on browser implementation and potential for large amounts of data loss. Therefore, when the client-side data is not available, we resort to reasonable approximation methods through server-side logging. Thus, we can reliably compute dwell time in a real world setting.

Client-Side Dwell Time: Client-side logging utilizes Javascript/DOM events [2] to record how users interact with the content story pages. Let us imagine the scenario demonstrated in Table 1 where the left column is a sequence of user interactions with a news article and the right column contains the corresponding client-side events, in the form of {event name, time stamp} tuples. In these events, `DOM-ready` indicates the ready-time of the body of the page, which can be considered as the start of the dwell time. `Focus` indicates that the user's focus was back on the body of the news article. `Blur` means that the article body lost the user attention. `BeforeUnload` is the time point immediately prior to the page being unloaded. Based on these events, we can compute dwell time on the client-side by simply accumulating time differences between `Focus` event and `Blur` events. From the above example, we have the dwell time as: $(t_3 - t_2) + (t_5 - t_4)$. We can clearly see that client-side approach can accurately capture users' actual attention even in multi-tabbed modern browsers. The major drawback of client-side logging is that it relies on the correctness of Javascript execution and on servers successfully receiving and logging client-sent data. Data loss in this client-server interaction can be very high, for example, because of loss in internet connection. In addition, users may also disable Javascript in their browsers.

[2]http://en.wikipedia.org/wiki/DOM_events

Table 2: Comparison of dwell time measurement. The first two columns are for `LE`, the middle two columns are for `FB` and the last two columns are for client-side logs. Each row contains data from a day.

#	DT. (LE)	#	DT. (FB)	#	DT. (C)
3,322	86.5	3,197	134.4	3,410	130.3
5,711	85.4	5,392	132.6	5,829	124.0

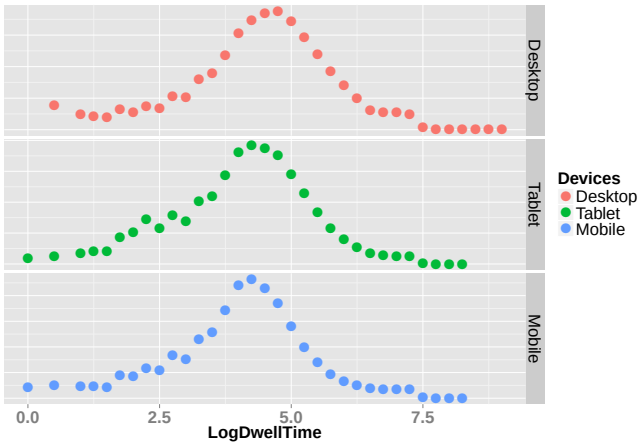

Figure 2: The (un)normalized distribution of log of dwell time for articles across different devices. The X-axis is the log of dwell time and the Y-axis is the counts (removed for proprietary reasons).

Server-Side Dwell Time: When client-side logs are not available, we resort to server-side logging to infer users' attention on item pages. The computation of dwell time on server-side are built on a number of heuristics. One approach is to simulate client-side user attention events by identifying pseudo `Focus` and `Blur` events from server logs. Consider the following sequence of logging events:

$$\{i, \texttt{Click}, t_1\} \rightarrow \{j, \texttt{Click}, t_2\} \rightarrow \{k, \texttt{Click}, t_3\}$$

where each event is a tuple of an item id, a event type and a timestamp. The dwell time for i and j can be computed as $t_2 - t_1$ and $t_3 - t_2$ respectively. A more complicated example is:

$$\{i, \texttt{Click}, t_1\} \rightarrow \{j, \texttt{Click}, t_2\} \rightarrow \{k, \texttt{Click}, t_3\} \rightarrow \{i, \texttt{Comment}, t_4\} \rightarrow \{n, \texttt{Click}, t_5\}$$

where the dwell time for page i can be computed as $(t_2 - t_1) + (t_5 - t_4)$. We denote this as `FB` (Focus/Blur) method. Another simpler heuristic is called `LE` (Last Event) method, which is to take the last event of the page as the *end-page* event and compute the interval of the first-event timestamp and the last one. From the example above, the dwell time of page i by `LE` would be $t_4 - t_1$. Both approximation methods have their own weaknesses: 1) The `FB` approach can *over-estimate* the dwell time because servers do not know the exact time the target story page loses its user attention. For the above example, if the last click happens on some other page, $(t_5 - t_4)$ interval could includes some user time spent outside the target page. The `FB` approach might also *under-estimate* the dwell time because servers also do not accurately know the time

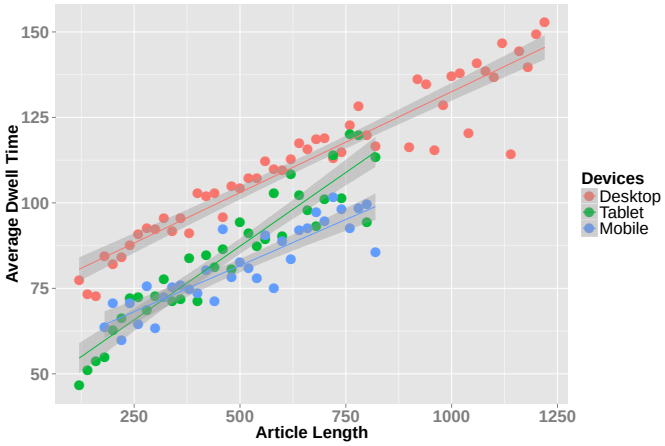

Figure 3: The relationship between the average dwell time and the article length where X-axis is the binned article length and the Y-axis is binned average dwell time.

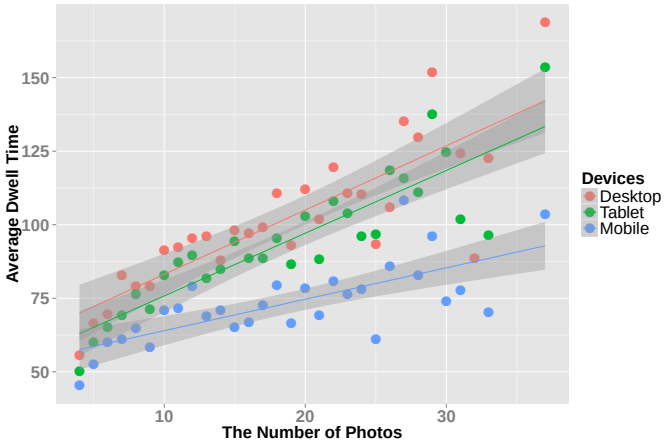

Figure 4: The relationship between the average dwell time and the number of photos on a slideshow where X-axis is the binned number of photos and the Y-axis is binned average dwell time.

the target page gains user attention. For the above example, if users have returned to the target page and read it for some additional time before commenting at t_4, the dwell time computation will not include the additional time. 2) The LE approach does not consider the scenario in which the users' reading focus could switch among multiple browser page tabs, thus over-estimating the dwell time. On the other hand, because the LE approach conservatively uses the last event on the target page to compute dwell time and servers do not know when the user abandons or closes the page (without the client-side *unload* event), it can also under-estimate the dwell time.

Because both approaches could over-estimate or under-estimate the item-level dwell time, we conducted a simple comparison study among FB, LE and the client-side logging. The results are shown in Table 2. The purpose of this study is to explore which server-side approach can be used to better approximate the client-side logging. We use two days' server-side logging events and client-side logging events for article pages, and compute the **average** dwell time by each method. Note that even for the same time period, different approaches use different sets of events to compute dwell time (see above example) and client-side events can be lost. Thus, the total number of articles considered varies (the first, the third and the fifth column). From the table, we can see that the average dwell time computed by the FB approach is very close to the client-side logging.Meantime, the LE approach greatly under-estimates the dwell time, compared with the client-side events. This result shows: through simulating users' reading attention switch events from server-side, the FB approach better handles item-level dwell time computation in multi-tabbed modern browser setting. Therefore, we now use FB as a relative reliable fall-back proxy to measure the item-level dwell time from server-side logging events when client-side logging is not available.

3.2 Dwell Time Analysis

In order to understand the nature of it, we analyze per-item per-user dwell time from a large real-world data collection from Yahoo. We plot the unnormalized distribution of log of dwell time in Figure 2. The data used for this figure is from one month's Yahoo homepage sample traffic. It is obvious that the log of dwell time follows a bell-curve. Many would guess the distribution of log of dwell time is a Gaussian distribution. However, Q-Q plot and also Shapiro–Wilk test [14] reject such an assertion. A further study of its formal distribution is in future work. Regardless of its normality, we observe that the bell-curve pattern holds for different time periods and different types of devices (see Figure 5 and 6, which we will discuss later).

Since dwell time approximates the time users spend on an item, it is natural to assume that given the same content quality, a longer news article would attract longer average dwell time across all users. In order to demonstrate this behavior, we investigate this issue on text heavy news article [3], and plot a scatter-plot of average dwell time per article versus article length in Figure 3 where X-axis is the length of article and the Y-axis is the average dwell time of that particular article from all users. In order to show things clearly, the dwell times and article lengths are binned into smaller buckets where each point represents a bucket. We show the scatter-plot of the dwell time against the length of the article on different devices, namely desktop, tablet and mobile devices. The black line is a fitted linear line for a particular device type with the 0.95 confidence interval in the grey area. From the figure, it is very clear that the length of the article has good linear correlation with the average dwell time across devices. Also, matching our intuition, the average dwell time on desktop is longer for long articles and the reading behavior on tablet and mobile devices are similar. Furthermore, the correlation becomes weaker when articles are very long: for desktop when the article is longer than 1,000 words, the plot has big variance; this indicates that users may have run out of their time-budget to consume the complete long story. Although the high correlation between the length of articles and average dwell time naturally leads to using the length of articles as a feature to predict average dwell time, we point out based on the observed data: (1) per-user dwell time (rather than binned average dwell time over all users) has little correlation with the article length; and (2) long dwell time may not necessarily reflect that users are really interested in the article. In other words, content length alone can hardly explain the per-user per-item dwell time, and we need to be careful of the bias of dwell time based user engagement measurements towards long length content stories. (We will revisit this issue in §3.4.)

[3]For other content types such as slideshow and video, the content length could be the number of slides in the slideshow and the video clip's raw duration, respectively.

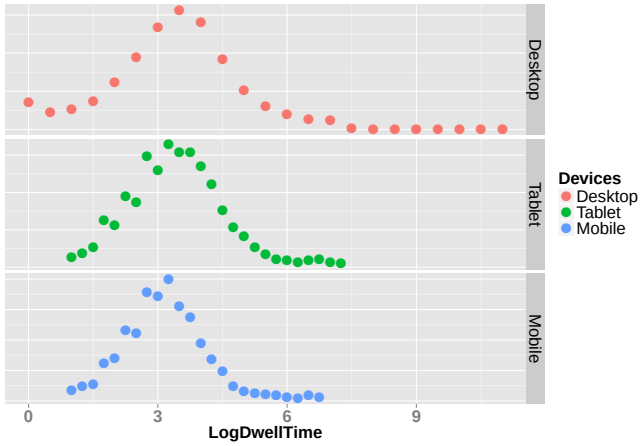

Figure 5: The (un)normalized distribution of log of dwell time for slideshows across different devices. The X-axis is the log of dwell time and the Y-axis is the counts (removed for proprietary reasons).

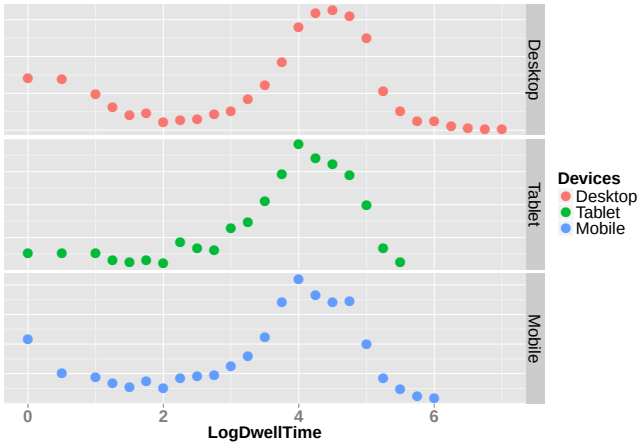

Figure 6: The (un)normalized distribution of log of dwell time for videos across different devices. The X-axis is the log of dwell time and the Y-axis is the counts.

For slideshows, a natural assumption would be that the larger the number of photos/slides, the longer the average dwell time these items would receive over all users. We demonstrate the relationships between the number of photos and the average dwell time on slideshows in Figure 4. Again, we binned the number of photos and the average dwell time. It is clear that the correlation is not as strong as the length of articles. For videos, we also observe the similar weak correlations between the duration of a video clips' and it's average dwell time.

3.3 Normalized Dwell Time

As may be obvious, users' consumption of content items varies by context. For example, in historical data, we found that users have on average less dwell time per article on mobile or tablet devices than on desktops. Also, users on average spend less time per slideshow than per article. Indeed, different content types, by their nature, would result in different browsing behaviors; thus we would expect different dwell times among these content types. In order to extract comparable user engagement signals, we introduce the normalized user dwell time to handle users' different content consumption behaviors on different devices for personalization. The technique discussed here can also be used to blend multiple content sources (e.g., slide-shows and articles) into a unified stream.

Although the distributions of users' per-item dwell time (from all users) for each content type is different, we found that each content type's distribution remains similar over a long time period. To demonstrate this observation, we further plot the log of dwell time of two important types of content: slideshows and videos in Figure 5 and Figure 6, respectively. Similar to the article case, we do not report the absolute values for both types. However, the patterns are again obvious. In all these cases, the log of dwell time has Gaussian-like distributions. Indeed, most of the dwell time distributions for each different content-type on different device platforms all surprisingly share the similar pattern. The same conclusion holds for different lengths of the time period. Also, we can easily see that the peak of log of dwell time is highest for videos, followed by articles and slideshows, which matches our intuitive understanding of these three types of content items.

Thus, the basic idea is, for each consumed item, we would like to extract its dwell time based user engagement level such that it is comparable across different context (e.g. content types, devices, instrumentations, etc.). We do this by normalizing out the variance of the dwell time due to differences in context. In particular, we adopt the following procedure to normalize dwell time into a comparable space:

1. For each content consumption context C, collect the historical per-item time spent data and compute the mean μ_C and standard deviation σ_C, both in log space.
2. Given a new content item i's time spent t_I in its context C_i, compute the z-value in log space: $z_i = \frac{\log(t_i) - \mu_{C_i}}{\sigma_{C_i}}$.
3. Compute the normalized dwell time of item i in the article space: $t_{i,\text{article}} = \exp(\mu_{\text{article}} + \sigma_{\text{article}} \times z_i)$.

In other words, all other types of items are now "comparable" after this transformation, and the normalized user engagement signals are then used for training recommendation models to handle different content types and can be deployed in different contexts.

3.4 Predicting Dwell Time

The average dwell time of a content item can be viewed as one of the item's inherent characteristic, which provides important average user engagement information on how much time the user will spend on this item. Predicting average dwell time for each content item can help labeling items when their dwell time are not available/missing. For example, content items that have never been shown to users (such as new items) will not have available dwell time. As another example, a user's dwell time on her clicked story may not be always be computed because there may be no subsequent server-side events from the same user. Therefore, leveraging predicted average dwell time can greatly improve the "coverage" (or alleviate the missing data issue). Not handling these situations, could degrade the effectiveness of applying dwell time in personalization applications. In this sub-section, we present a machine learning method to predict dwell time of article stories using simple features.

The features we consider are topical category of the article and the context in which the article would be shown (e.g., desktop, tablet or mobile). We use Support Vector Regression (SVR)[4] models to predict dwell time. The model is trained from a sample of user-article interaction data. We show the features and their corresponding weights in Table 3. Most features are categorical and we use $\log(\text{Dwell Time})$ as the model response. We can loosely in-

[4]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Table 3: Features and corresponding weights for predicted dwell time. The features are shown in the order of magnitude of weights. The left column shows positive weights and the right negative weights.

Name	Weight	Name	Weight
Desktop	1.280	Apparel	-0.001
Mobile	1.033	Hobbies	-0.010
Tablet	0.946	Travel & Tourism	-0.039
Content Length	0.218	Technology	-0.040
Transportation	0.136	Environment	-0.065
Politics	0.130	Beauty	-0.094
Science	0.111	Finance	-0.151
Culture	0.100	Food	-0.173
Real Estate	0.088	Entertainment	-0.191

terpret the weights of these features as how much that feature contributes to the article's average dwell time prediction. The feature weights match our **current** expectation for average users' article reading behavior: longer articles can lead to higher predicted average dwell time; people spend a longer time reading articles on desktop devices than mobile devices; more serious topics can lead users to dwell longer. Potentially, the predicted average dwell time could be leveraged to normalize the dwell time-based user engagement signal (as discussed in §3.3); however, this is non-trivial as the interplay between the dwell time features and users' experience is not obvious. For example, will recommending more serious topics that have long average dwell lead to better or worse user experience? We will leave answering this question for future work.

4. USE CASE I: LEARNING TO RANK

In this section, we investigate how to leverage item-level dwell time to train machine-learned ranking (MLR) models for content recommendation.

The Basic MLR Setting: In traditional MLR, a query q is represented as a feature vector $\boldsymbol{q}$ while a document d is represented as a feature vector $\boldsymbol{d}$. A function g takes these two feature vectors and outputs a feature vector $\boldsymbol{x}_{q,d} = g(\boldsymbol{q}, \boldsymbol{d})$ for this query-document pair (q, d). Note that g could be as simple as a concatenation. Each query-document pair has a response $y_{q,d}$, in traditional IR, which is usually the relevance judgment. Typically this judgment is common to all users, that is, there is no user-specific personalization. Depending on the particular paradigm (e.g., point-wise, pair-wise or list-wise), a machine learned model imposes a loss function l which takes one or all documents belonging to a query q as the input, approximating the individual relevance judgment, pair-wise relevance preferences or the whole list ordering. In the context of content recommendation, we can simply borrow the idea of MLR by treating user interests as queries and articles (or other types of items) as documents. Although this formulation looks promising, there are two challenges. One is how to construct a feature vector for queries (users) and the second is how to utilize user activities to infer relevancy between users and documents. The discussion of the first question is out of this paper's scope. Here, we focus on the second question. While the definition of relevance judgments might be unambiguous in IR, it is not straightforward in the context of content personalization. One cheap and easy approach is to use users' click-through data as relevance judgments. Essentially, in such case, we use $y_{d,u} = \{0, 1\}$, a binary variable, to indicate whether an article d (the "document" in IR setting) is clicked by the user u (the "query" in IR setting). Under this formalism, a MLR model indeed optimizes (CTR).

In this paper, we use the Gradient Boosted Decision Tree (GBDT) algorithm [6] to learn the ranking functions. GBDT is an additive regression algorithm consisting of an ensemble of trees, fitted to current residuals, gradients of the loss function, in a forward stepwise manner. It iteratively fits an additive model as:

$$f_t(x) = T_t(x; \Theta) + \lambda \sum_{t=1}^{T} \beta_t T_t(x; \Theta_t) \tag{1}$$

such that a certain loss function $L(y_i, f_T(x_i))$ (e.g., square loss, logistic loss) is minimized, where $T_t(x; \Theta_t)$ is a tree at iteration t, weighted by a parameter β_t, with a finite number of parameters Θ_t, and λ is the learning rate. At iteration t, tree $T_t(x; \beta)$ is induced to fit the negative gradient by least squares. That is:

$$\hat{\Theta} = \arg\min_{\beta} \sum_{i}^{N} w_i (-G_{it} - \beta_t T_t(x_i); \Theta)^2 \tag{2}$$

where w_i is the weight for data instance i, which is usually set to 1, and G_{it} is the gradient over the current prediction function: $G_{it} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{t-1}}$. The optimal weights of tree β_t are determined by $\beta_t = \arg\min_\beta \sum_i^N L(y_i, f_{t-1}(x_i) + \beta T(x_i, \theta))$. More details about GBDT, please refer to [20]. As mentioned above, if we use click/non-click as responses, we simply treat $x_i = \boldsymbol{x}_{q,d}$ and $y_i = y_{d,u}$. In fact, all previous research on MLR-based content recommendation system has been focusing on using click-based information for training and evaluation. For example, Bian et al. [2] and Agarwal et al. [1] have used users' click/view data in Today module in Yahoo for optimizing CTR for content recommendation.

Dwell Time for MLR: There are two intuitive ways to incorporate dwell time into MLR frameworks. Let γ_d be the average dwell time for article d. Taking the GBDT algorithm mentioned above, we could have: 1) Use the per-article dwell time as the response, treating $y_i = h(\gamma_d)$ and 2) Use the per-article dwell time as the weight for sample instances, treating $w_i = h(\gamma_d)$ where the function h is a transformation of the dwell time. In both cases, we promote articles that have high average dwell time and try to learn models that can optimize for user engagements. In all our experiments, we found that $h = \log(x)$ yields the best performance.

We show the effectiveness of MLR model firstly from an offline experiment. We use data from a bucket of traffic of a Yahoo property and split it uniformly at random into training and test sets, using a 70-30 split. We repeat this sampling multiple times and the average results across all train-test splits are shown in Table 4. The first observation is that either method of using dwell time as learning target or instance weight can improve three major ranking metrics. The second observation is that, dwell time as an instance weight leads to the best performance. We further validate these findings in online buckets, shown in Figure 7. Without disclosing the absolute numbers, we show the same three buckets with respect to two types of performance metrics: 1) CTR (shown on the top) and 2) a user engagement metric(shown on the bottom). The user engagement metric is a proprietary one, which can be explained as the quality of users' engagement with Yahoo homepage's content stream. Each data point represents the metric on a particular day. We report the bucket metrics for a three month period between June 2013 and August 2013. Initially, the three buckets were running the same linear model and we can see from the first three data points (three-day data), both CTR and the user engagement metric are similar. Then, we update the models as follows: 1) A: a linear model optimizes click/non-click, 2) B: a GBDT model optimizes click and

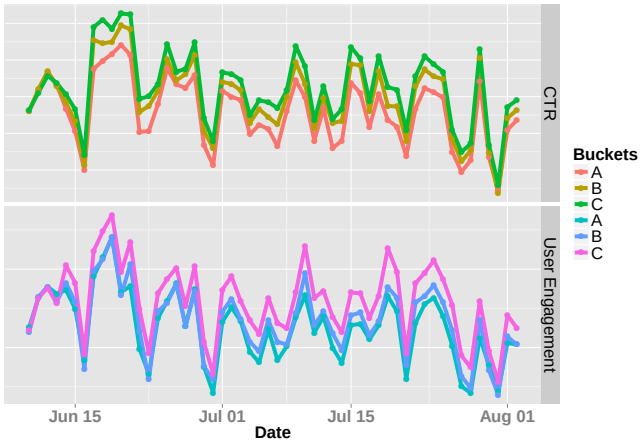

Figure 7: The relative performance comparison between three buckets. The top figure shows the relative CTR difference and the bottom figure shows the relative user engagement difference.

Table 4: Offline Performance for Learning to Rank

Signal	MAP	NDCG	NDCG@10
Click as Target	0.4111	0.6125	0.5680
Dwell Time as Target	0.4210	0.6201	0.5793
Dwell Time as Weight	0.4232	0.6226	0.5820

3) C: a GBDT model optimizes dwell time $h(\gamma_d)$. It is interesting that after updating models, we observe a divergence of the performances. Bucket A becomes the worst performing bucket in both metrics while bucket C outperforms other buckets consistently for almost two months time until all buckets were ended. Therefore, from our empirical experiences, optimizing dwell time not only achieves better user engagement metrics but also improves CTR as well. One plausible cause is when optimizing towards dwell-based engaging signals rather than high CTR, users may better like the content recommended, come back to the site and click more. We leave the thorough analysis of this finding as future work.

5. USE CASE II: COLLABORATIVE FILTERING

In the previous section, we elaborated how dwell time can be used in the context of MLR, requiring enough per-item interaction activities from a user to build usable interest profile. Thus, it is very challenging for the previous approach to recommend content to users who are not very active (e.g. new users), or to recommend new content that has no overlap with the users' historically read items. CF techniques utilize user engagement signals on particular stories to discover users who have similar reading interest from a broad audience and target them with the same or similar content. In this section, we validate how dwell time can be used in CF frameworks to better improve the performance.

We formalize the problem into the classic matrix factorization framework where latent features of items and user latent features are jointly learnt from a user-item interaction matrix. Matrix factorization models [9] have provided state-of-the-art performance in many CF problems. In prior work, MF models were developed to operate on discrete user ratings (e.g., movie ratings) as labels, thus making it difficult to directly apply these models in settings where explicit ratings are usually missing or hard to collect. For instance, users come to Yahoo's homepage to consume news items such as articles, videos and slideshows by only browsing and clicking on particular items that they are interested in without providing any explicit rating feedback, even though the user interface allows users to "like" or "dislike" the item. Therefore, users' feedback under this context is implicit.

In this paper, we propose to use dwell time of users rather than asking them to give ratings or using click information as implicit rating feedback. User feedback is represented as a $M \times N$ sparse matrix I, where M is the number of users, and N is the number of items, and each entry in the matrix is one user feedback, which is denoted as $r_{i,j}$. For dwell time, $r_{i,j} \in R$ or $[0, 6]$ for normalized dwell time; for click/view, $r_{i,j} \in [0, 1]$. Formally, we aim to predict the unobserved entries in the matrix based on the observed data. Rank-based matrix factorization is used. We decompose the sparse matrix I as U and V, to minimize the following objective function:

$$\arg\min_{U,V} \sum_{i=1}^{M} \sum_{r_{i,j}<r_{i,k}} U_i(V_j - V_k)^T + \lambda(|U|_2 + |V|_2) \quad (3)$$

where $r_{i,j}$ refers to positive examples, $r_{i,k}$ refers to negative examples. The difference between the training process of dwell time and click is the bootstrap process of negative examples. For click, no-click feedbacks are treated as negative data. For dwell time, for each click with dwell time, we randomly sample another click feedback with less dwell time or non-click feedback.

Dataset: The data used in this experiment is collected from a bucket (a small sample traffic) of a Yahoo property, spanning a three month period. We perform training on the first three months and prediction on last month. We further remove the users and items with less than 10 clicks. In this dataset, we have $147,069$ distinct users and $11,535$ distinct items, yielding $4,358,066$ events in the training set and $199,420$ events in the test set.

Evaluation Method and Metrics: We group users by time periods and construct sessions of all items a user consumed in a particular time period (e.g., months, day, hour and etc.). Instead of evaluating how well we can predict clicks/non-clicks, we evaluate the proposed methods in terms of ranking metrics. We use Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) as main evaluation metrics. We start the definition of metrics by focusing on a particular user. Let Precision@k for a particular user in one session as: $\texttt{Prec@k} = \frac{1}{k}\sum_{j=1}^{k} r_j$ where k denotes the truncation position and r_j is whether the ranked document at the j position is relevant or not. Average Precision, the measure using two levels of relevance judgement, is depended on the basis of Precision $\texttt{AP} = \frac{1}{|D^+|}\sum_j r_j \times \texttt{Prec@j}$ where $|D^+|$ denotes the number of relevant items with respect to the user in this session. Given a ranked list for a session, we can compute an AP for this session. Then MAP is defined as the mean of AP over all sessions for all users. NDCG@k is a measure for evaluating top k positions of a ranked list using multiple levels (labels) of relevance judgement. It is defined as $\texttt{NDCG@k} = N_k^{-1}\sum_{j=1}^{k} \frac{2^{r_j}-1}{\log_2(1+j)}$ where NDCG is just for all positions.

Experimental Results: There are two different strategies for evaluation. The first experiment takes the data from first three months as training data while the last one as test data. The second one is to use sliding window to perform daily or weekly prediction. The overall performance is shown in Table 5 where the table is split into three parts, the first part about monthly prediction, the middle part about weekly prediction and the bottom part about daily prediction. For weekly prediction and daily prediction,

Table 5: Performance for Collaborative Filtering

Performance for Monthly Prediction			
Signal	**MAP**	**NDCG**	**NDCG@10**
Click as Target	0.3773	0.7439	0.7434
Dwell Time as Target	0.3779	0.7457	0.7451
Performance for Weekly Prediction			
Signal	**MAP**	**NDCG**	**NDCG@10**
Click as Target	0.6275	0.5820	0.5813
Dwell Time as Target	0.6287	0.5832	0.5826
Performance for Daily Prediction			
Signal	**MAP**	**NDCG**	**NDCG@10**
Click as Target	0.6275	0.5578	0.5570
Dwell Time as Target	0.6648	0.5596	0.5589

the metrics are averaged numbers across multiple weeks or days. We also vary the latent dimension K for both click version and the dwell time version and only report the best performance across different K values. We can observe that in all evaluation methods and all metrics, the model to optimize dwell time has a comparable performance as the one to optimize clicks. In addition, the performance of dwell time based models are consistently better than the click based ones. One plausible reason, for the small overall improvement, is that content features or user-side information may be needed for better predicting dwell time based rating. Deeper analysis and experimentation on the benefit of dwell time based `CF` models is future work.

6. DISCUSSION AND CONCLUSIONS

In this paper, we demonstrated how dwell time is computed from a large scale web log and how it can be incorporated into a personalized recommendation system. Several approaches are proposed for accurately computing item-level user content consumption time from both client side and server side logging data. In addition, we exploited the dwell time distributions of different content types for normalizing users' engagement signals into the same space. For `MLR`, we proposed using per-user per-item dwell time as the learning target and demonstrated that it can result in better performances. For `CF`, we used dwell time as a form of implicit feedback from users and demonstrated how it can be incorporated into a state-of-the-art matrix factorization model, yielding competitive and even better performances than the click-optimized counterpart. For future work, we would like to design dwell time based user engagement metrics and explore how to optimize these metrics directly. We would also like to investigate better ways to normalize dwell time. This will enable us to extract better user engagement signals for training recommendation systems thereby optimizing for long term user satisfaction.

7. REFERENCES

[1] D. Agarwal, B.-C. Chen, P. Elango, and R. Ramakrishnan. Content recommendation on web portals. *Communications of the ACM*, 56(6):92–101, June 2013.

[2] J. Bian, A. Dong, X. He, S. Reddy, and Y. Chang. User action interpretation for online content optimization. *IEEE TKDE*, 25(9):2161–2174, Sept 2013.

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

[4] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of ICML*, pages 129–136, 2007.

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

[6] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis - Nonlinear methods and data mining*, 38(4):367–378, Feb. 2002.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM*, pages 263–272, 2008.

[8] Y. Kim, A. Hassan, R. W. White, and I. Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of WSDM*, pages 193–202, 2014.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[10] H. Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011.

[11] C. Liu, J. Liu, N. Belkin, M. Cole, and J. Gwizdka. Using dwell time as an implicit measure of usefulness in different task types. *Proceedings of the American Society for Information Science and Technology*, 48(1):1–4, 2011.

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

[13] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI*, pages 452–461, 2009.

[14] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika 52 (3-4)*, pages 591–611, 1965.

[15] R. W. White and D. Kelly. A study on the effects of personalization and task information on implicit feedback performance. In *Proceedings of CIKM*, pages 297–306, 2006.

[16] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of SIGIR*, pages 391–398, 2007.

[17] S. Xu, H. Jiang, and F. C. M. Lau. Mining user dwell time for personalized web search re-ranking. In *Proceedings of IJCAI*, pages 2367–2372, 2011.

[18] S. Xu, Y. Zhu, H. Jiang, and F. C. M. Lau. A user-oriented webpage ranking algorithm based on user attention time. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, pages 1255–1260, 2008.

[19] P. Yin, P. Luo, W.-C. Lee, and M. Wang. Silence is also evidence: Interpreting dwell time for recommendation from psychological perspective. In *Proceedings of SIGKDD*, pages 989–997, New York, NY, USA, 2013. ACM.

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