User Action Interpretation for Personalized Content Optimization in Recommender Systems

Anlei Dong
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
anlei@yahoo-inc.com

Jiang Bian
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
jbian@yahoo-inc.com

Xiaofeng He
Microsoft Search Technology Center Asia
Microsoft China Lt.
xihe@microsoft.com

Srihari Reddy
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
sriharir@yahoo-inc.com

Yi Chang
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
yichang@yahoo-inc.com

ABSTRACT

User interaction plays a vital role in recommender systems. Previous studies on algorithmic recommender systems have mainly focused on modeling techniques and feature development. Traditionally, implicit user feedback or explicit user ratings on the recommended items form the basis for designing and training of recommendation algorithms. But user interactions in real-world Web applications (e.g., a portal website with different recommendation modules in the interface) are unlikely to be as ideal as those assumed by previously proposed models. To address this problem, we build an online learning framework for personalized recommendation. We argue that appropriate user action interpretation is critical for a recommender system. The main contribution in this paper is an approach of interpreting users' actions for the online learning to achieve better item relevance estimation. Our experiments on the large-scale data from a commercial Web recommender system demonstrate significant improvement in terms of a precision metric over the baseline model that does not incorporate user action interpretation. The efficacy of this new algorithm is also proved by the online test results on real user traffic.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experiments

Keywords

action interpretation, content optimization,personalization

1. INTRODUCTION

Digital content publishers, including portal websites, such as MSN and Yahoo!, have started providing Web users with a wide range of modules of Web content in a timely fashion. As shown in Figure 1, the portal Yahoo! presents today's emerging events, news of various aspects, and trending queries from a search engine to Web users. However, Web users usually have short attention spans while facing the explosion of Web content, referred as *information overload*. Therefore, it is necessary for those Web publishers to optimize their content by identifying the most relevant content to attract and retain users to their site on an ongoing basis.

An effective automatic recommendation system becomes indispensable for serving the users who visit portal websites, and personalization is also a desirable feature since it can further tailor content presentation to suit individual's interests rather than take the traditional "one-size-fits-all" approach.

In this paper, we propose a *parallel-serving-buckets* online learning framework which can instantaneously model the user actions, i.e. browse and click, in the recommender system so as to better serve the users. We use a dedicated model to estimate the relevance score for each candidate content item, which is specified as estimating the click-through rate (CTR) in our online learning framework. To achieve personalization, we employ a divide-and-conquer strategy which categorizes users into diverse groups based on their interests as modeled by user action information.

Our main contribution is: an approach of interpreting users' actions for the online learning to achieve better item relevance estimation, by taking into consideration the factor of user engagement.

The rest of this paper is organized as follows. In Section 2, we propose our online learning framework for recommendation and point out the critical challenge for achieving personalization and reaching better performance. Section 3 proposes appropriate user action interpretation to refine our online learning approach which can result in better performance recommendation. A large-scale evaluation and discussion based on the data from a commercial portal website are presented in Section 4, including testing the models on real user traffic. We conclude the paper in Section 5.

2. PERSONALIZED RECOMMENDATION

In this section, we introduce our recommender system framework and its components, which can achieve effective personalization through online learning. Furthermore, we point out the critical

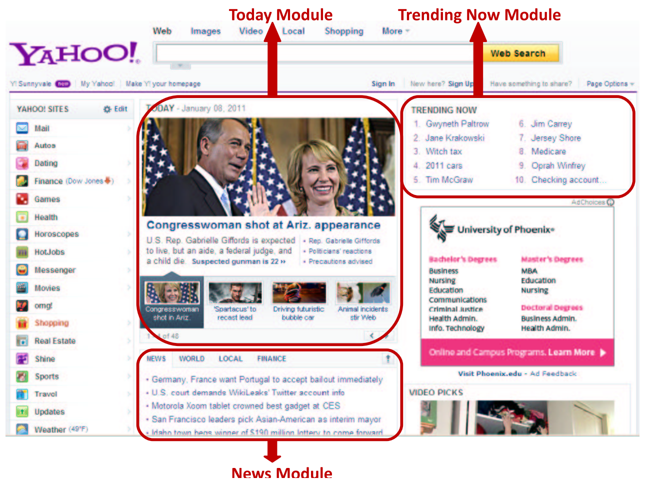

Figure 1: A snapshot of Yahoo! front page. The page contains multiple recommendation modules such as *Today module*, *Trending Now module* and *News module*.

challenges for building such a system, which will be addressed in detail in the next section.

2.1 Problem formulation

We target recommendation applications of web portal frontpages. Our goal is to optimize content recommendation such that a certain user engagement metric, such as overall click-through rate (CTR), is maximized. For a pool of candidate items, human editors can be employed to manually rank the candidate items according to content attractiveness and users' interests and then recommend top ranked items to users. However, it requires expensive human effort and cannot guarantee that the most attractive and relevant items are recommended to users due to the interest gap between editors and Web users. Therefore, we attempt to design a recommender system that achieves content recommendation by automatically estimating candidate items' attractiveness and relevance to users' interests. Such a recommender system has three critical characteristics:

1. *Online learning*. To attract more users to browse and click content displayed on the portal frontpage, an online learning methodology is necessary because it enable us to model users' behaviors (i.e. clicks and views) on the frontpage as implicit feedback and adapt the recommendation model in real time (or almost real time) to user feedback, so as to serve more relevant content to users and provide better recommendation optimization. Figure 2 illustrates the flowchart of this online learning approach. To achieve better recommendation, it becomes essential to employ an effective method for updating per-item models. In this paper, we employ an *Estimated Most Popular* (EMP) model ([1]) to estimate CTR.

2. *Per-item model*. For each candidate item in the recommender system, we use a dedicated model to estimate its attractiveness/relevance score. We then rank all items by their respective scores in the descending order and display the top ranked ones to users. Under the scenario of online learning where real-time user feedbacks are available, the ranking score of an item can be estimated by its click-through rate (CTR), which represents a strong signal of attractiveness of this item to users. In the rest of this paper, we will apply the per-item model by default without explicit declaration.

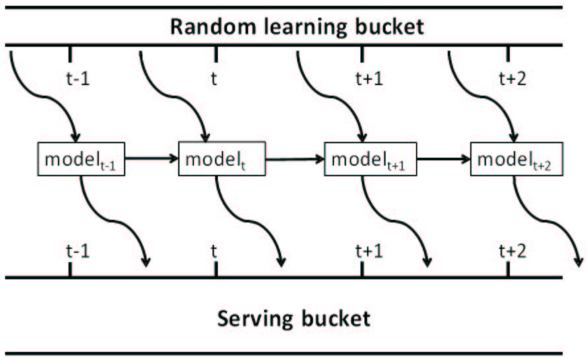

Figure 2: Online learning flowchart. A Random learning bucket is used for exploration purpose. At the end of each time interval, the model for each candidate item is updated based on the users' clicks and views in random learning bucket during this time interval. In the next time interval, the updated model is applied to the corresponding candidate item in the serving bucket. In this way, all the candidate items are displayed by ranking scores (computed by their corresponding updated models) in the serving bucket.

3. *Personalization*. We use a personalization approach to provide users with a personalized experience of highly attractive and relevant content, so as to enhance user engagement, conversions, and long-term loyalty. A *divide-and-conquer* strategy is employed to achieve personalization: we divide users into a few different groups based on user profiles; for each group of users, the recommender system serves them with the models which are updated using only the feedback information created by the users belonging to the same group. This approach is called *user segmentation*.

2.2 Challenge

In this section, we have presented the online learning approach for our recommender system which implements personalization based on user segmentation. However, there is a critical challenge for implementing this recommender system:

Within each group, how to best utilize user feedback information to achieve effective online learning? The learning samples provided for online learning are characterized based on user feedback actions (i.e., clicks and views). However, after conducting user segmentation, learning samples in each user segment are not as ample as using all samples, therefore, correct understanding of user actions becomes more critical for obtaining effective recommendation models.

3. USER ACTION INTERPRETATION

We argue that good understanding and exploitation of user actions, including clicks and views, can effectively benefit the modeling by better understanding users' general interests and the items' attractiveness to users.

As shown in Figure 1, there is usually more than one content recommendation module on portal website. Different content modules are likely to compete with each other on a densely packed interface such as the frontpage of the portal website. Therefore, one user visit on the portal website may not necessarily mean the user is really engaged in all the content modules displayed to the user. Here, engagement means that the user examined or at least partly examined recommended content. For example, when a user visits the

Yahoo! frontpage as shown in Figure 1, it is possible she totally ignores the displayed Trending Now module contents as she may be attracted by the contents of other modules such as Today module, or she directly goes for other services such as search and e-mail.

For a recommendation module, accurate CTR estimation should be based on the events where users were really engaged in this module, instead of all the events where the contents of this module were merely displayed to the users. In our work, we identify three categories of events regarding user engagement:

1. **Click event**: *click event* is an event where the user clicked one or more items in the module after she opened the web page. In a click event, the user is engaged in the module under study because she must have examined at least some of items recommended by the module. Obviously, click events is useful for CTR estimation.

2. **Click-other event**: *click-other event* contains at least one action on other application/modules in the interface (such as clicking items displayed by other modules, doing search in search box, etc). Obviously, click-other events should be excluded from being used for CTR estimation.

3. **Non-click event**: besides click events and clicks-other events, there are also *non-click events* in which users had no action such as click or search after they opened the web page. For a non-click event, unlike click event or click-other event, it is not straightforward to determine whether or not the user actually examined the module under study as usually the system cannot track user's eyes. However, based on user's historic behaviors, it is still possible to deduce if the user intends to examine the module or not. Intuitively, if a user often clicked the module under study in the past, it implies this user is interested in this module so that it is likely she actually examined the module in the latest event. For a user, we can check the number of clicks on the module during a specified length of past period and use such click number to present the prior probability that this user actually examined the module in the event.

4. EXPERIMENTS

4.1 Setup

4.1.1 Data Set

To validate our proposed approaches, we conduct experiments on the data from a real-world content recommendation module, i.e. the Trending Now module on the Yahoo! frontpage (as shown in Figure 1). We collected events in terms of *views* and *clicks* from a *random learning bucket* of the Trending Now module during ten days from November 30th, 2010 to December 9th, 2010. The pool of candidate items may change multiple times during each day. To protect privacy, all the users are anonymized in this dataset.

In the *random learning bucket*, candidate queries are randomly selected and displayed to users at all of positions with equal chances. An event records a user's action on the served content items (i.e. trending queries) on the Trending Now module, which is either "*view*" or "*click*". More specifically, we represent each event e as a set of tuples:

$$e = \langle u, t, p, q_p, a \rangle, \; p = 1, 2, \ldots, 10$$

where u denotes the user, t represents the time stamp for this event, q is the served query, p denotes the position at which the trending query q is displayed (there are ten positions on the Trending Now module), a represents the action which is either *view* or *click*. Note that one logged *view* event e only means that the query q was displayed to the user u, who did not necessarily examine the query. In the whole dataset, there are totally hundreds of millions of events with multiple millions of unique users.

Table 1: An illustrative example for evaluation metric (precision) computation. For an actual event in the random learning bucket, Item 1 was ranked at Position 1 and clicked by the user. For Model 1: $precision_1$ = 1, $precision_2$ = 1, $precision_3$ = 1, and so on. For Model 2: $precision_1$ = 0, $precision_2$ = 0, $precision_3$ = 1, and so on. Model 1 is regarded as a better model as the clicked item is ranked higher by Model 1.

actual ranking	predicted ranking by Model 1	predicted ranking by Model 2
1 (clicked)	**1**	2
2	5	3
3	4	**1**
4	3	5
5	2	4

4.1.2 Evaluation Metrics

Before we apply a new model to the production system with real user traffic, we need to evaluate different candidate models by some offline experiments. In our offline experiments, we simulate the online learning procedure as illustrated in Figure 2: all per-item models at time t are updated based on the click/view samples during the latest 5-minute interval $[t-1, t]$ in the *random learning bucket*; the updated models are then applied to the candidate items during the next time interval $[t, t+1]$ so that we use the item ranking predicted by the updated models. For the clicks that actually happened during $[t, t+1]$ in the *random learning bucket*, the evaluation metric is computed by comparing these actual clicks with the predicted ranking. Intuitively, a good modeling approach should lead to high correlation between the actual clicks and the predicted ranking.

More specifically, for those clicks that actually happened at Position 1 in the *random learning bucket*, we define $precision_i$ as the number of the clicked items that are ranked at Position from 1 to i by the model prediction. Table 1 illustrates two examples for such precision computation. Note that the reason we only use the clicks at Position 1 for evaluation is that the clicks on other positions should be counted with more weight due to position bias, so it is more straightforward to use clicks at Position 1 for evaluation. To protect business-sensitive information, we report only relative precision, instead of precision itself.

In the following experiments, we evaluate different methods for obtaining the per-item models on the Trend Now dataset described in Section 4.1.1. By following the online learning simulation procedure during the 10-day period, the overall precision values are computed by aggregating the precision of recommendation in each of the 5-minute time intervals.

4.2 Effects of user engagement

We now explore effects of different types of user engagement on our online learning approach. As shown above, we have evaluated the recommendation performance if using only *click events* for model learning. In this section, we will first analyze the effects of *click-other events* by measuring the performance of the recommendation model which excludes all *click-other events* for model learning. Table 2 demonstrates the relative precision gains when training without *click-other events* over training on original whole

Table 2: Relative precision gain when training without *click-other event* over training on the original whole dataset.

Model	$prec_1$	$prec_2$	$prec_3$	$prec_4$	$prec_{10}$
EMP-kmeans	11.11%	7.05%	8.22%	7.70%	5.46%

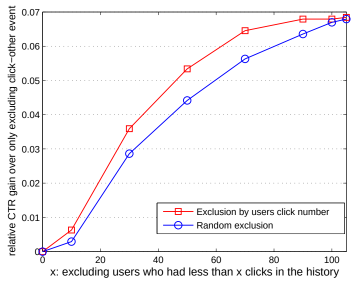

Figure 3: Relative precision gain by various user segmentation methods over the baseline, conducted on the original dataset.

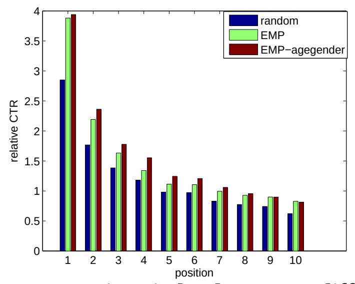

Figure 4: CTR comparison in bucket test at different positions.

dataset. This table clearly shows that excluding *click-other events* can improve CTR estimation.

As discussed in Section 3, besides *click events* and *click-other events*, there are also *non-click events* in which users had no action after they opened the Yahoo! frontpage. Unlike *click events* or *click-other events*, it is not straightforward to determine whether *non-click events* are useful for CTR estimating. We hypothesize that the utility of such event highly depends on the corresponding user's previous engagement on the module. The intuition is that if a user previously had higher *click* engagement on a content module, it is more likely she will examine the module in a future event.

To validate our hypothesis, we evaluate the performance of **EMP-kmeans** by gradually excluding the *non-click events* based on the number of clicks the corresponding user generated during a specified length of time. Figure 3 illustrates the precision values at Position 1, compared with another **EMP-kmeans** method which randomly excludes the same number of users. Note that, in this experiment, we have pre-processed the training data by removing all *click-other events*, and we will never exclude any user who has ever engaged in a *click events*. From this figure, we can find that gradually excluding *non-click events* based on users' previous number of clicks performs better than random exclusion, which indicates that the history of users' previous engagement is a good signal to judge the users' potential engagement in future events.

4.3 Bucket test results

Based on the simulation results we previously presented, we also online-test two models on real users. The two models are the **EMP** model (baseline) and the **EMP-agegender** model. Regarding learning samples for model update, **EMP** model uses all events while **EMP-agegender** model uses only click events. Now we will compare them by bucket test. We divide users into 3 different buckets. The first bucket is our random learning bucket, which only occupies a small amount of user traffic. The other two buckets are serving buckets, which rank queries by the **EMP** model and the **EMP-agegender** model respectively, and display the queries to the users within the corresponding buckets. These two serving buckets

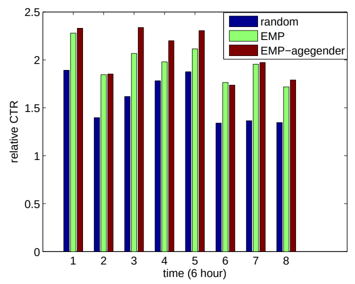

Figure 5: CTR comparison in bucket test at 8 consecutive 6-hour time slots during 2 days.

have similar amounts of user traffic. We run the bucket test over two days, and we compare the CTRs of the two buckets. Overall, the CTR in the **EMP-agegender** model bucket is 6.01% higher than the **EMP** model bucket.

We also compare CTRs at different positions and different times, as shown in Figure 4 and Figure 5 respectively. To protect confidential information, we do not show absolute CTRs; instead, we do scaling on all CTRs by a constant scaling factor. Therefore, we can observe the relative CTRs in different buckets. In Figure 4 and Figure 5, we first observe that the random learning bucket always has lowest CTRs. This is not surprising since the purpose of this bucket is only for exploration.

In terms of the CTR improvement of the **EMP-agegender** model over the **EMP** model, Figure 4 shows that the most significant improvement comes from the top positions while the improvement is less at lower positions. This trend verifies that compared with the **EMP** model, the **EMP-agegender** model presents queries at top positions which are more relevant to the users.

For the two days' test period, we divide the 48 hours into 8 consecutive 6-hour time slots. In Figure 5, we compare the CTRs of different buckets within each of these time slots. We observe that at different time slots, the improvements by the **EMP-agegender** model are different. The reason is that the content pool is changing over time. During some time slots, there are very popular items that are attractive to all users. In this case, the personalization model the **EMP-agegender** is not so effective. During other time slots, the candidate items are more appropriate for personalization so that the **EMP-agegender** model is more effective.

5. CONCLUSION

In this paper, we have studied a few important topics towards user action interpretation in recommender systems. We build a personalized content optimization system using *parallel-serving-buckets* framework. In this framework, we introduce action interpretation for improving online learning. We explore the effect of user engagement in the online learning procedure. Large-scale evaluations demonstrate that we can significantly improve the performance of the recommender system by integrating these user action interpretation factors.

6. REFERENCES

[1] D. Agarwal and B.-C. Chen and P. Elango, Spatio-Temporal models for estimating click-through rate. In *Proc. of the 18th International World Wide Web Conference (WWW)*, 2009.