User Action Interpretation for Online Content Optimization

Jiang Bian, Anlei Dong, Xiaofeng He, Srihari Reddy, and Yi Chang

Abstract—Web portal services have become an important medium to deliver digital content and service, such as news, advertisements, etc., to Web users in a timely fashion. To attract more users to various content modules on the Web portal, it is necessary to design a recommender system that can effectively achieve online content optimization by automatically estimating content items' attractiveness and relevance to users' interests. User interaction plays a vital role in building effective content optimization, as both implicit user feedbacks and explicit user ratings on the recommended items form the basis for designing and learning recommendation models. However, user actions on real-world Web portal services are likely to represent many implicit signals about users' interests and content attractiveness, which need more accurate interpretation to be fully leveraged in the recommendation models. To address this challenge, we investigate a couple of critical aspects of the online learning framework for personalized content optimization on Web portal services, and, in this paper, we propose deeper user action interpretation to enhance those critical aspects. In particular, we first propose an approach to leverage historical user activity to build behavior-driven user segmentation; then, we introduce an approach for interpreting users' actions from the factors of both user engagement and position bias to achieve unbiased estimation of content attractiveness. Our experiments on the large-scale data from a commercial Web recommender system demonstrate that recommendation models with our user action interpretation can reach significant improvement in terms of online content optimization over the baseline method. The effectiveness of our user action interpretation is also proved by the online test results on real user traffic.

Index Terms—Action interpretation, Content optimization, Personalization, Recommender Systems

1 Introduction

Recent years have witnessed rapid growth of the Internet, which has become an important medium to deliver digital content to Web users instantaneously. Digital content publishers, including portal websites, such as MSN (http://msn.com/) and Yahoo! (http://yahoo.com/), and homepages of news media, like CNN (http://cnn.com/) and the New York Times (http://nytimes.com/), have all started providing Web users with a wide range of modules of Web content in a timely fashion. For example, as shown in Figure 1, there are various specific content modules on the Yahoo! portal, such as *Today Module* presenting today's emerging events, *News Module* presenting news of various aspects, and *Trending Now Module* presenting trending queries from a search engine that is used in the portal website. Although there are multiple content venders and plenty of content, Web users usually have short attention spans while browsing the portal. Therefore, it is necessary for those Web publishers to optimize their delivered content by identifying the most attractive content to catch users' attention and retain them to their portal sites on an ongoing basis.

Often, human editors are employed to manually select a set of content items to present from a candidate pool. Although editorial selection can prune low-quality content items and ensure certain constraints that characterize the portal website, such human effort is quite expensive and usually cannot guarantee that the most attractive and personally relevant content items are recommended to users especially when there is a large pool of candidate items. As a result, an effective and automatic content optimization becomes indispensable for serving users with attractive content in a scalable manner. Personalization is also a desirable feature for the content optimization since it can further tailor content presentation to suit an individual's interests rather than take the traditional "one-size-fits-all" approach.

In general, personalized content recommendation on portal websites involves a process of gathering and storing information about portal website users, managing the content assets, analyzing current and past user interactive actions, and, based on the analysis, delivering the right content to each user. Traditional personalized recommendation approaches can be divided into two major categories: *content-based filtering* and *collaborative filtering*. In the former method, a profile is generated for a user based on content descriptions of the content items previously rated by the user. However, the main drawback of this approach is its limited capability to recommend content items that are different than those previously rated by users. Collaborative filtering, which is one of the most successful and widely used techniques, analyzes users' ratings to recognize commonalities and recommend items by leveraging the preferences from other users with similar tastes. However, since portal websites usually aim at recommending emerging information, such as news in the News Module and the Today Module and trending queries in the Trending Now Module on Yahoo! as shown in Figure 1,

- *J. Bian is with Yahoo! Labs, Sunnyvale, CA 94089. E-mail: jbian@yahoo-inc.com*
- *A. Dong is with Yahoo! Labs, Sunnyvale, CA 94089. E-mail: anlei@yahoo-inc.com*
- *X. He is with Microsoft, Beijing, China. E-mail: xihe@microsoft.com*
- *S. Reddy is with Google, Mountain View, CA 94043. E-mail: sriharirh@gmail.com*
- *Y. Chang is with Yahoo! Labs, Sunnyvale, CA 94089. E-mail: yichang@yahoo-inc.com*

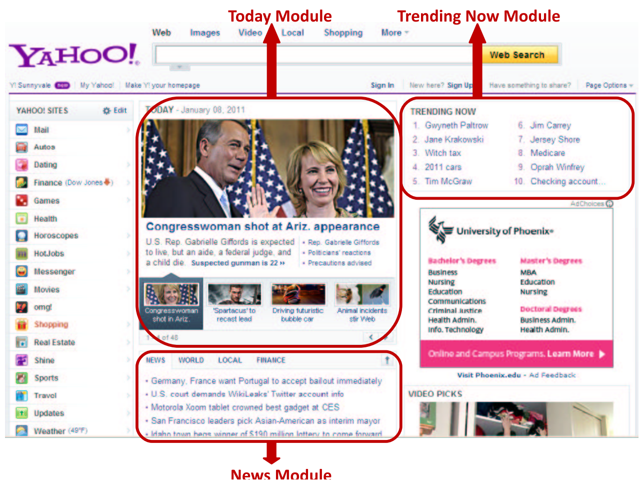

Fig. 1. A snapshot of Yahoo! front page. The page contains multiple recommendation modules such as *Today module*, *Trending Now module* and *News module*.

collaborative filtering may not be appropriate since it suffers from the cold-start problem [26].

Although hybridization can alleviate some of the weaknesses associated with collaborative filtering and other recommendation techniques, there are still a few critical challenges that are unique to content optimization at portal websites and have not been well-studied in the literature. First, as there is a big traffic of users' visiting in every minute, portal websites can attract a large number of users actions in terms of browsing and clicks on presented content modules. Such user action information can obviously provide strong signals of users' recent interests on the content item. However, it is quite a challenge to incorporate them into the recommendation model in real time or near real time. Second, as the purpose of personalization is to provide users with personalized experience of highly attractive content, the problem of how to appropriately define user segments (i.e. divide users into different groups according to their interests) to achieve personalization becomes crucial for effective content optimization. Moreover, since user action information plays a vital role in modeling users' interests and content items' attractiveness in the recommender system, accurate understanding of user actions become one of essential factors to reach high recommendation performance.

To address all of these challenges, in this paper, we first introduce a *parallel-serving-buckets* online learning framework which can instantaneously model the user actions (i.e. browse and click) in the recommender system in order to serve users with better recommendation. In this online learning framework, we propose to use a dedicated model to estimate the attractiveness score, specified as click-through rate (CTR), for each candidate content item individually. To achieve personalized recommendation in our framework, we employ a divide-and-conquer strategy, which first categorizes users into diverse groups based on their interests as modeled by user action information, and then serve users in each group with recommendation modeled by user actions of those users in the same group. Furthermore, to build effective recommendation model, we propose more accurate approaches to interpret user actions, especially in terms of user engagement and position bias when users browse or click content items. The new user action interpretation can benefit recommendation by sampling those user visit events that are really informative to learning the recommendation model.

To summary, the main contributions in this paper include:

- An effective online learning framework for taking advantage of user actions to serve content recommendation in real time or near real time;
- A new approach to leverage historical user activity to build a behavior-driven user segmentation, which results in higher engagement after application to the personalized content optimization;
- A novel approach of interpreting users' actions for the online learning to achieve better estimation on content items' attractiveness, including taking into account the factors of both user engagement and position bias.

The rest of this paper is organized as follows. Section 2 reviews the literature in content optimization. In Section 3, we propose our online learning framework for recommendation and point out the critical challenges for achieving better recommendation. To address these challenges, Section 4 proposes more accurate user action interpretation to achieve better personalization and refine the online learning approach by filtering out non-informative user visit events. A large-scale evaluation and discussion based on the data from a commercial portal website are presented in Section 5. In this section, we also include the model tests on real user traffic beyond offline experiments. We conclude the paper and point out future work in Section 6.

2 RELATED WORK

Content optimization is defined as the problem of selecting content items to present to a user who is intent on browsing for information. There are many variants of the problem, depending on the application and the different settings where the solution is used, such as articles published on portal websites [3], [2], news personalization [12], [28], recommendation of dynamically changing items (updates, tweets, etc), computational advertising [7], [32] and many others. This work will address one variant that displays the best set of trending queries from a search engine in a module on the portal website. This application is different from the task of query suggestion in web search in the sense that it recommends popular queries to users from a certain pool of globally trending queries while query suggestion suggests queries relevant to what the user just submitted to a search engine.

There are two major categories of approaches for content recommendation, content-based filtering and collaborative filtering. The former one reflects the scenario where a recommender system monitors a document stream and pushes documents that match a user profile to the corresponding user. Then, the filtering system uses explicit relevance feedback from users to update the user's profile using relevance feedback retrieval models [41], [40], [42] or machine learning algorithms [27], [39]. Collaborative filtering goes beyond merely using document content to recommend items by taking advantage of information from other users with similar tastes and preferences [26], [21], [20], [19], [35]. Previous studies [31], [38], [14], [25] have tried to combine both techniques for more effective content optimization.

Most of existing studies focus on building the offline recommendation model. However, in this work, we aim at addressing the problem of online content recommendation. Those previous approaches may not be good enough because, in the context of online content recommendation, both the content pool and users' interests change very frequently, and offline models cannot be updated according to such changes very efficiently. To address this problem, we propose an online learning framework for personalized recommendation. In our work, we also leverage user behavior information to combine the two techniques. In particular, we apply user action interpretation to model relevance feedback used in content-based filtering. We employ user behavior based segmentation, which follows the direction of collaborative filtering, to improve the effectiveness of the content recommendations. Note that, in this work, the content itself of the items is not used in building the recommendation model since the corresponding computational cost is a little large especially when we target at online recommendation. But, the content based features could be used to address the cold-start problem even in online recommendation. Due to the space limit, we will not cover this direction in this paper.

To improve personalized content optimization, using historic user behavior information on the respective applications have been explored by a couple of previous studies. These have demonstrated that such information can be extremely helpful for improving recommendation performance. For instance, many studies propose building user profiles for content scoring in the recommender system. [6] describes an approach to build user profile models for adaptive personalization in the context of mobile content access. YourNews [5] allows users to customize their interest profiles through a user model interface. These studies on user behaviors show the benefit from customization, but also warns of the downside impact on system performance. In our application we take advantage of user behavior information without explicitly soliciting it from users. Newsjunkie [16] provides personalized news feeds for users by measuring news novelty in the context of stories the users have already read. In our work, we go beyond the module of trending queries, the target applications. We also utilize user behavior information from other modules on the portal website to optimize the recommendation performance.

A personalized service may not be exactly based on individual user behaviors. The content of the portal website can be tailored for a pre-defined audience, based on offline research and conjoint analysis. In very early studies [37], homogeneous groups of consumers are entailed by the use of a priori segmentation. For example, recommendations can be based on demographic classes categorized by users' personal attributes. However, such user segmentation on the basis of simple demographic variables does not necessarily reflect different users' interests on the content of the portal website. [10] and [11] recently proposed user behavior feature-based models for personalized services at individual and segmentation levels, respectively. Those personalized models are shown to outperform several demographic segmentation models. However, they did not analyze the quality of each of more than 1000 user behavior features. In our work, we take advantage of user click information to select a subset of user behavior features with high quality.

Considerable research on user action interpretation has been conducted in the context of web search. In particular, online user behavior modeling has attracted much attention in recent years. Some work discussed user behavior models based on controlled user studies [22], [33], while other studies focused on large-scale log analysis [36], [15]. Recently, some research [18], [17] has used eye-tracking studies to understand in detail how searchers examine search results, meanwhile dwell time interpretation has also attracted significant attention [24], [23] and has been extensively used for various information retrieval tasks [8], [29], [4]. However, user action interpretation has not received much attention in the studies of content optimization. Our work proposes to deeply analyze user action interpretation for content optimization. In particular, we leverage user behavior information to sample training examples in order to remove those that can benefit little for learning the effective model. To our best knowledge, there are very few of previous works that have studied interpreting user actions in the context of content optimization. Das et al. [13] and Liu et al. [30] have made earlier effort to enhance news recommendation based on users' click behaviors. Beyond them, our work will propose a more comprehensive study on the effects of users' behaviors for online content optimization, and our study will be expanded into any content module

3 Online Learning for Personalized Recommendation

In this section, we introduce our online learning framework for personalized content recommendation as well as the key components of this framework. Furthermore, we point out the critical challenges for building such a system, which will be addressed in detail in the next section.

3.1 Problem formulation

As we target recommendation applications at content modules on web portals, our goal is to optimize content recommendation such that a certain user engagement metric, such as overall 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 items displayed on the content modules on portal websites, an online learning methodology is necessary because it enable us to model users' behaviors (i.e. clicks and views) on the portal websites as implicit feedbacks and update the recommendation model accordingly in real time (or almost real time), so as to serve more attractive and relevant content to users.
2) *Per-item model*. To build effective online recommendation model, the straightforward but reliable method is to apply a dedicated model for each candidate content item to estimate its attractiveness/relevance score. Using these dedicated per-item models, we can rank all items by their respective recommendation scores in the descending order and present 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 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.
3) *Personalization*. Personalization has become very important for content optimization as it provides users with a customized experience of highly attractive and relevant content, so as to enhance user engagement, conversions, and long-term loyalty. To introduce personalization for content optimization, our online learning framework employs a *divide-and-conquer* strategy. In particular, 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 user actions only by those belonging to the same group. This approach is referred as personalization driven by *user segmentation*.

In the rest part of this section, we will elaborate each of these three components in details.

3.2 Online Learning

To enable online learning for content optimization, we introduce a *parallel-serving-buckets* approach. Figure 2 illustrates the flowchart of this online learning approach for the system. We use the term *bucket* to denote a part of the whole users visit traffic on portal websites. Different *buckets* yield different strategies to serve recommendation. Specifically, in our *parallel-serving-buckets* framework, we divide the whole users visit traffics into two parallel buckets serving simultaneously in the system: *random learning bucket* and *serving bucket*. When a user visits the portal website, this visit event can be randomly assigned into either the *random learning bucket* or the *serving bucket*.

Within the *random learning bucket*, a certain number of items are randomly sampled from the pool of candidates to serve as recommended items for each user visit. In our system, we limit the *random learning bucket* to occupy only a small fraction of the whole traffic, thus, the probability that a user visit falls into this bucket is very small. We use this *random learning bucket* to estimate item CTRs for all time intervals. Although serving candidate items at random is obviously not the optimal recommending strategy, this *random learning bucket* can benefit the online learning in another way. In particular, since each of items from the candidate pool have the equal chances to be served to users in the *random learning bucket*, we can obtain the unbiased estimated CTR for each item based on users' feedbacks in this bucket. Such unbiased CTRs can be further used as strong signals of users' interests on the respective items to benefit online learning model, i.e. the model of *serving bucket*.

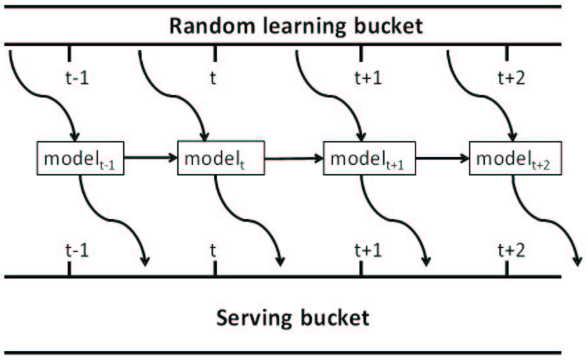

Fig. 2. Online learning flowchart. A Random learning bucket is used for exploration purpose. At the end of each time interval, the model for each candidate item is updated based on the users' clicks and views in random learning bucket during this time interval. In the next time interval, the updated 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.

In our *parallel-serving-buckets* approach, as shown in Figure 2, all the models in both buckets are updated simultaneously every 5 minutes (i.e., the time interval $[t, t+1]$ equals 5 minutes in Figure 2). In general, within the *serving bucket*, each per-item model, at a certain time point $t+1$, is adapted to the observations (users views and clicks) the corresponding item from the *random learning bucket* during the time interval $[t, t+1]$. The updated models are then applied to the candidate items in *serving bucket* and the items are displayed by the ranking scores in descending order.

3.3 Per-Item Model

To build effective online recommendation model, the straightforward but reliable method is to apply a dedicated model for each candidate content item to estimate its attractiveness/relevance score. Using these dedicated per-item models, we can rank all items by their respective recommendation scores in the descending order and present the top ranked ones to users. To adapt per-item model in our online learning framework, it is essential to employ an effective method for updating per-item models. In this paper, we employ a *Estimated Most Popular* (EMP) model. Assume during the time interval $[t, t+1]$, an item was displayed to users $n_{[t,t+1]}$ times, which resulted in $c_{[t,t+1]}$ clicks, and assume this item's CTR estimation is p_t predicted by its previous model at time t; then, for the model of this item at time $t+1$, the CTR estimation of this item is updated as

$$p_{t+1} = \frac{\gamma_t p_t + c_{[t,t+1]}}{\gamma_t + n_{[t,t+1]}} \tag{1}$$

where γ_t is the sample size of the prior belief, which is updated as

$$\gamma_t = \omega \gamma_{t-1} + n_{[t-1,t]} \tag{2}$$

in which ω is a time-decay factor to discount the samples which happened long time ago.

The intuition in Equation (1) is that, given its prior probability p_t, the CTR estimation is updated according to new observations of clicks and views during time interval $[t, t+1]$. The sample size γ_t is used to control the balance between the prior probability and new observations. The higher the value of γ_t, the more confidence we have on the prior probability. If the value of γ_t is set lower, the CTR estimation relies more on the new observations. More details of EMP can be found in [1]. In this paper, the EMP approach is regarded as the baseline, which is to be compared with new proposed approaches based on user action interpretation that we will explore in the next section.

3.4 User Segmentation

To introduce personalization for online content optimization, we propose employing *user segmentation* based approach, in which homogeneous groups of users are entailed by a priori segmentation [37], where each segment of users are served with the dedicated recommendation model. There are a few other categories of personalization approaches; however, the user segmentation approach yields advantages in terms of both simplicity and reliability, especially for real-world commercial recommender systems. Note that, user segmentation based personalization is not a deep personalization as it cannot provide the specific solution for each user. But, the following experiments based on such shallow personalization still demonstrate that there is a big potential benefit from personalization to improve online content optimization.

To integrate user segmentation into the online learning approach as introduced in Section 3.2, users are divided into a small number of groups, each of which has its exclusive online learning and serving process. In other words, each user group has its own per-item models which are learned based on clicks and views only from the users belonging to the corresponding group, and the serving results using these models are also only applicable to the users belonging to the corresponding group.

To obtain this user segmentation, we propose generalizing a set of user features and then applying clustering techniques to group users based on extracted features. We collect two major categories of user features that is available to the portal website owner: 1) *Explicit features*: the personal information explicitly requested by the portal website, such as age, gender, location, preferences, etc. 2) *Implicit features*: various types of users' behaviors tracked by portal website, such as browsing and purchasing patterns of users on the pages within this website, etc. Both of these two categories of user features can implicitly represent users' preferences and recent interests over Web content. In our recommender system, each user is represented as a vector of features, whose dimensionality can be more than 1,000 in our experiments. The specific clustering techniques will be discussed in the next section.

3.5 Challenges

In this section, we have presented the online learning approach for content optimization which implements personalization based on user segmentation. However, there are two critical challenges:
1) *How to appropriately divide users into different groups?* The criterion for a good user segmentation is that homogeneous users (users with similar interests, characteristics, behaviors, etc.) should belong to the same group, while heterogeneous users should belong to different groups. Although it is possible to heuristically set some rules to group users based on explicit user features, it may not be optimal to represent users' different interests. Since we can extract a large number of implicit user behavior features, which provides more signals of users' interests, it becomes challenging to develop an automatic method for deriving better user grouping based on users' diverse interests, because the high dimensionality of the user features may easily lead to over-fitting if they are directly used by machine learning algorithms.
2) *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.

In the next section, we will address both of these two challenges by introducing appropriate user action interpretation, the effectiveness of which will be demonstrated in Section 5.

4 User Action Interpretation

To address those challenges raised in Section 3.5, we argue that good understanding and exploitation of user actions, including clicks and views, can effectively benefit the recommendation by better understanding users' general interests and the items' attractiveness to users. In the following of this section, we first propose to use historical user click information to select discriminant features for user segmentation (Section 4.1). Then, we study how to improve online learning based on more accurate interpretation of users' actions in terms of clicks and views with considering both user engagement and click position bias (Section 4.2).

4.1 Action Interpretation for User Segmentation

To appropriately divide users into different segments, the most straightforward method is to group uses based on their explicit static features, such as demographic information (Section 4.1.1). However, this heuristic rule-based method may not be optimal since the generated segmentation is ad hoc and it ignores large amounts of implicit user behavior information which can better reflect users' interests. We propose to take advantage of the rich user behavior information, especially the history of users' clicks on the portal website, to obtain a user segmentation that results in a better serving content optimization. In particular, we will introduce two different clustering techniques for leveraging such important information.

4.1.1 Segmentation by Demographic Information

Intuitively, users with the same demographic features, such as age and gender, are likely to have the similar interests. Accordingly, the straightforward approach for user segmentation is to group users based on combination of several demographic features which are provided as profiles to the portal website by users themselves. In this paper, we apply both *age* and *gender* and group users into 7 segments, as illustrated in Table 1.

TABLE 1
User segmentation based on demographic features.

Segment	Age Range and Gender
f-u20	$10 <$ age $<= 20$, gender = female
f-u40	$20 <$ age $<= 40$, gender = female
f-u80	$40 <$ age $<= 80$, gender = female
m-u20	$10 <$ age $<= 20$, gender = male
m-u40	$20 <$ age $<= 40$, gender = male
m-u80	$40 <$ age $<= 80$, gender = male
unk	unknown age or gender

Note that, besides the information of age and gender, there are also other useful demographic features to divide users, such as location. We have conducted some primitive experiments, which also illustrate that user segmentation based on location can also benefit the recommendation performance. Due to the limited space, we will not show the details in this paper.

Overall, this heuristic segmentation approach is simple and easy to implement; however, it may not necessarily be optimal since 1) the demographic information users filled as profiles on the portal website may be noisy or fictitious, and 2) such segmentation may not be fine-grained enough for the purpose of personalized recommendation. In the following of section, we will propose to leverage user behavior information, which we believe can be a better indication for users' interests on the Web, to build a user segmentation so as to better serve content optimization.

4.1.2 Unsupervised Clustering

When users surf on the Web, there is plenty of information about user's behavior on the content displayed to them. Although the interactions between the user and the content items vary depending on the types of content items involved, we can always observe or generalize some behavioral patterns from user side. For a content item posted on a Web page, a user may click to see more details. Based on the log of users' actions on a commercial portal website, we can extract several thousand binary features describing users' behavior patterns. Such rich user behavior information can provide implicit signals for indicating users' interests which can improve the performance of personalized content optimization.

Intuitively, users with similar behavior patterns are more likely to have the similar interests. Thus, we can define a feature vector for each user by using those binary features. However, due to the large number of binary features from the commercial portal website, we first attempt to reduce the dimension of user features by doing feature selection. The straightforward method is to select features based on *support*, which means the number of samples having the feature. Only the features of high support above a prefixed threshold, e.g. 5% of the population, will be selected.

In this paper, we propose another feature selection method by utilizing users' click behavior on the module served by our recommender system. In particular, we first select a set of items which have been clicked by users in this content module during a certain period. Then, we can generate a feature vector of each item by aggregating the feature vectors of users who ever clicked the item in the certain period. After that, we normalize each dimension of the feature vector across different items. Finally, we can select those feature dimensions whose respective highest normalized value is above a prefixed threshold. We consider that this new feature selection method can be more effective since the selected ones are more important to those users who have more engagement on the certain content module rather than the whole set of users.

After selecting a set of important behavior based features, we can define each user in this new feature space and then apply unsupervised clustering method, e.g. k-means, to cluster users. The clustering output will form the segmentation for users.

4.1.3 Tensor Segmentation

A more sophisticated approach [11], tensor segmentation, has demonstrated its effectiveness for conjoint analysis, which is a method in market research to measure how customers with different preference value different features of the product or service. Since tensor segmentation is a scalable conjoint analysis technique to learn the user preference in the presence of user features and product characteristics, by viewing content items as the product in conjoint analysis, we can apply this technique for user segmentation in our study.

The basic idea of the tensor segmentation is as follows. Denote a user by a user feature vector x_i, a content item by an item feature vector z_j, then a tensor product of the user, x_i, and the content item, z_j, is defined as

$$s_{ij} = \sum_{a}^{|z_j|} \sum_{b}^{|x_i|} w_{ab} x_{i,b} z_{j,a}.$$

It can be simplified as vector-matrix multiplication as

$$s_{ij} = x_i^T W z_j,$$

where W is a $|x_i|$ by $|z_j|$ matrix, which is also known as a bilinear model and has been extensively studied in the literature [9], [34]; s_{ij} represents the indicator of the correlation between the user and the item, which can be conveniently related to the response, r_{ij} (i.e. click or not), of the user x_i on the content item z_j by logistic regression as

$$p(r_{ij}|s_{ij}) = \frac{1}{1 + e^{(-r_{ij} s_{ij} + \iota)}}$$

where ι is the global offset to deal with highly imbalanced click/view events.

In the solution of [11], user-specific bias μ_i and query-specific bias γ_j were introduced. Thus the tensor indicator s_{ij} is transformed into

$$\hat{s_{ij}} = s_{ij} - \mu_i - \gamma_j.$$

we can easily obtain the matrix W by solving the logistic regression problem. After matrix W is available, any user x can be projected to new feature space as $W^T x$, a vector with length of $|z_j|$. In the new feature space with the same dimension as content items, a clustering algorithm can be applied to the transformed user feature vector to obtain the user clusters. We utilize the K-means algorithm on transformed user feature vectors to generate user clusters.

Note that, in contrast to traditional unsupervised clustering techniques, the tensor segmentation is a supervised method in the sense that it takes advantage of users' responses (i.e. click or not) to various content items as labels to infer users' interests under the semantic feature space of content items.

4.2 Action interpretation for online learning

As introduced in Section 3.2, the online learning algorithm relies heavily on user clicks and views, which are critical for developing effective content optimization. For a candidate item, its CTR is estimated based on the number of clicks and views for this item (Equation 1), which implies that correct interpretation of user actions is important since click/view samples are derived from the user actions logged by the portal website. Along this direction, we address two important factors in this section, including *user engagement* and *position bias*.

4.2.1 User Engagement

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:

- **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. Note that one click event consists of the user's click on one item and her views on other items along with it. Obviously, click events is useful for CTR estimation.
- **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.
- **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.

4.2.2 Position Bias

For an item, we attempt to aggregate its clicks/views at all positions for CTR estimation; otherwise, the samples at a single position may not be enough for reliable estimation. As introduced in Section 3.2, item CTR estimation is based on the click/view counts by Equation (1), in which all the clicks are treated equally. However, click position should also be taken into consideration.

In the example of the Yahoo! Trending Now module, there are always ten recommended queries that are dis-

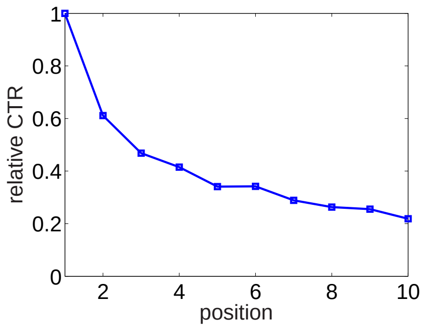

Fig. 3. Relative CTRs at different positions in random learning bucket.

played to users (shown in Figure 1). When an item is displayed at different positions, the probability that it will be clicked is different. Figure 3 illustrates this position bias effect in random learning bucket (not serving bucket). We collect random learning bucket data (Section 3.2) from the Yahoo! Trending Now module during a period of one month, and compute average CTR values at different positions. For the purpose of confidential information protection, we only show relative CTR values, which are obtained by dividing CTR values with the CTR value at Position 1. We observe that moving from top to bottom (Position 1, 2, 3, . . ., 10), the CTR values drops monotonically. Note that the candidate queries in the random learning bucket are randomly displayed at any position. Therefore, the CTR variation at different positions reflects the fact that an item's click probabilities is affected by position.

There are at least two factors that may lead to such position bias: 1) an item displayed at different positions may have different probabilities of being examined by users; 2) if a user examines an item at bottom positions, the probability that she clicks this item is lower than the case that this item is displayed at top positions. This is because when the item is displayed at bottom positions, users may have less confidence that this item is high quality. We call this phenomenon *position decay factor*. More specifically, for an item that is displayed to the user at Position j, the probability that the user will click this item is

$$P(\text{click} \mid \text{pos} = j) = \alpha_j P(\text{exam} \mid \text{pos} = j) P(\text{click} \mid \text{exam}) \tag{3}$$

where $P(\text{click} \mid \text{exam})$ is the probability that the item is clicked if it is examined by the user and it is position independent. $P(\text{exam} \mid \text{pos} = j)$ is the probability that the item is examined by the user if it is displayed at Position j and α_j is the position decay factor. As it is difficult to decouple the two factors α_j and $P(\text{exam} \mid \text{pos} = j)$, we re-write as

$$P(\text{click} \mid \text{pos} = j) = \beta_j P(\text{click} \mid \text{exam}), \tag{4}$$

so that only β_j is position dependent, which we call the *position prior factor*. Usually the closer the position to bottom, the lower the value of β, as shown in Figure 3.

It is obvious that the CTR estimation in Equation 1 is for the position-independent CTR $P(\text{click} \mid \text{exam})$. Therefore, in Equation (1), when a click at Position j is to be incorporated into the model, we need to overweight this click sample by $\frac{\beta_1}{\beta_j}$. To estimate relative position prior factor $\frac{\beta_1}{\beta_j}$, we can accumulate a period of historical data in the random learning bucket as we did for Figure 3, then use the average CTR values at different positions to compute:

$$\frac{\beta_1}{\beta_j} = \frac{\overline{\text{CTR}_1}}{\overline{\text{CTR}_j}}. \tag{5}$$

where $\overline{\text{CTR}_j}$, the average CTR at Position j is the approximation of $E_{q,t}(\text{CTR}_{q,t,j})$.

5 EXPERIMENTS

In this section, we design experiments to validate our proposed approaches, i.e., action-interpretation-based user segmentation (Section 4.1) and online learning with considering action interpretation, including user engagement and position bias (Section 4.2), can improve the performance of content optimization. We first describe the datasets from a commercial portal website and evaluation metrics in Section 5.1. Then, we report the offline results of a large-scale evaluation for our proposed approaches in Section 5.2, Section 5.4 and Section 5.3. In Section 5.5, we introduce our online test results on real user traffic. This is also called *bucket test*, in which different models are applied simultaneously to real user visit traffic, but they are all limited in small parts of the whole traffic, respectively. Such *bucket test* enable us to compare performance of different models by observing their respective recommendation results in different small parts of user visit traffic. Finally, we take further discussion in Section 5.6.

5.1 Setup

5.1.1 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.

As introduced in Section 3.2, 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:

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

where u denotes the user, t represents the time stamp for this event, q is the served query, $p(q)$ 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*. As discussed in Section 4.2, one logged *view* event e only means that

TABLE 2
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, $precision_4$ = 1, and $precision_5$ = 1; while for Model_2: $precision_1$ = 0, $precision_2$ = 0, $precision_3$ = 1, $precision_4$ = 1, and $precision_5$ = 1. 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

the query q was displayed to the user u, who did not necessarily examine the query. In the whole dataset, there are totally several hundreds of millions of events with multiple millions of unique users.

5.1.2 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 2 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 5.1.1. 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.

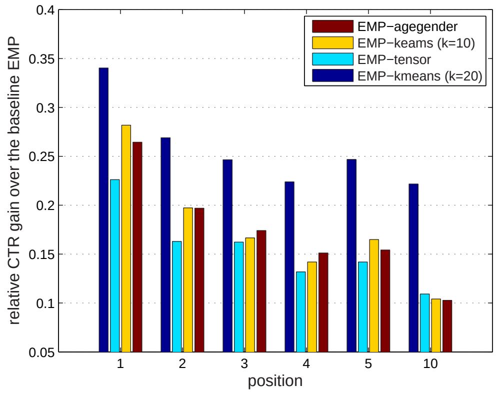

Fig. 4. Relative precision gain by various user segmentation methods over the baseline, conducted on the original dataset.

5.1.3 Methods Compared

To evaluate the effectiveness of user segmentation boosted content optimization, we compare the following methods:

- **EMP** (baseline, Section 3.2): In this method, we adopt estimated most-popular model without any user segmentation.
- **EMP-agegender** (Section 4.1.1): In this method, we generate a user-segmentation-based on age-gender features. We group users into 7 segments, as illustrated in Table 1. For each segment, we train separate EMP models.
- **EMP-kmeans** (Section 4.1.2): In this method, we apply unsupervised clustering, i.e., k-means, to group users based on their behavior features, and the feature set is selected based on feature values normalized across all candidate trending queries. For each user segment, we adopt separate EMP models. In the following experiments, we test results by setting k as 10 and 20, respectively.
- **EMP-tensor** (Section 4.1.3): In this method, we employ the tensor segmentation technique to group users into different segments. Then, we train separate EMP models for each user segment. In our experiment, the number of segments is set to 10 for this method.

In the following of this section, we conduct the experiments on the dataset described in Section 5.1.1. The model learning will be based on either all the events, or only on click events. Recall in Section 4.2.1, a click event means the user clicked one or more items in the module in this event, which implies user's high engagement.

5.2 Effects of user segmentation

To learn user segmentation (Section 4.1) for EMP-kmeans and EMP-tensor, we use another older dataset with fourteen days (from November 1st, 2010 to November 14th, 2010) to select user features as described the beginning of Section 4.1.2.

Figures 4 and 5 demonstrate the relative precision gain of various user segmentation methods over the baseline EMP model, conducted on the original dataset and on the dataset with only *click events*, respectively. From these

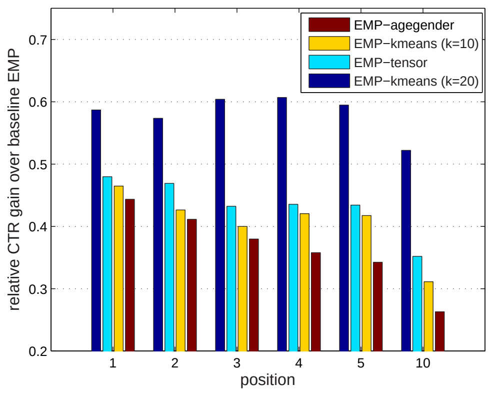

Fig. 5. Relative precision gain by various user segmentation methods over the baseline, conducted on the dataset with only *click events*.

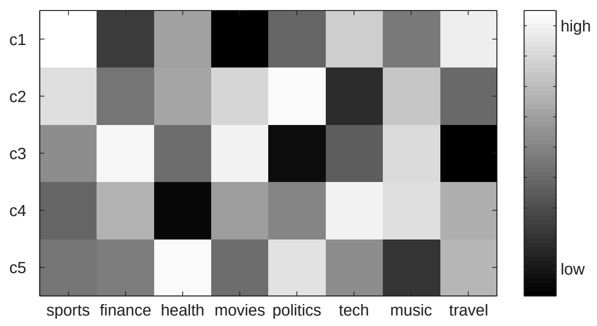

Fig. 6. User segments' preferences on selected item topics in the five example user segments. Each square's gray level indicates the preference of a segment on the corresponding topic, from white (*like*) to black (*dislike*).

figures, we can find that all the user segmentation methods can outperform the baseline model on both datasets. Furthermore, the user behavior-driven segmentation methods, **EMP-kmeans** and **EMP-tensor**, can reach much better performance than **EMP-agegender** where user segmentation is based on static demographic features. And, when the number of segment is set to 20, **EMP-kmeans** can increase the performance significantly over **EMP-kmeans** and **EMP-tensor** with number of segments set to 10. We expect that **EMP-tensor** will perform much better if we use 20 segments during training. However, due to the tight dependency on human editorial data as well as high computational cost at the learning stage , we determine **EMP-kmeans** is a practical method estimating CTR. Thus, in the following experiments, we will employ **EMP-kmeans** with 20 segments as our user segmentation method by default.

To further explore the effects of the level of personalization, we perform a study on its influence on the performance gain by user segmentation methods. In particular, we evaluate the performance of personalized recommendation against varying number of user segments. Figure 7 reports the relative precision gains of **EMP-kmeans** over the baseline EMP model with varying number of user segments,

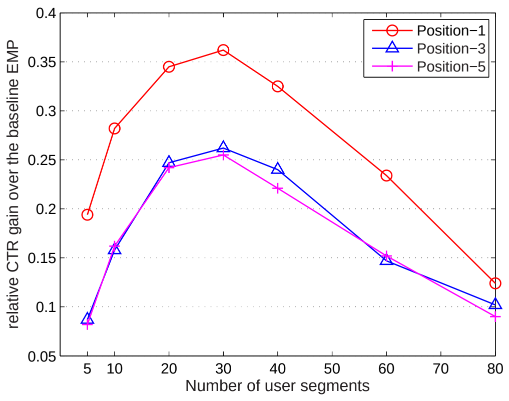

Fig. 7. Relative precision gain of EMP-kmeans over the baseline against varying number of user segments.

which is conducted on the original dataset. From this figure, we can see that the performance of **EMP-kmeans** increases when the number of user segment raises from 5 to 30; however, if the number of user segment keeps increasing, the performance of **EMP-kmeans** will decline. Such result indicates that higher level of personalization can give rise to increasing performance for online recommendation; nevertheless, when the number of user segment keeps increasing, the amount of learning samples for each user segment becomes sparse, which will hurt the performance of the learned recommendation model conversely.

To visualize the characteristics of different user segments generated by **EMP-kmeans** when k is set as 20, we utilize the centroid of each user segment in the feature space as a representative to illustrate each segment's preference on item topics. The centroids in the space of item topics are presented in Figure 6 as a heat-map. The gray level indicates users' preferences, from *like* (white) to *dislike* (black). We only show five example user segments and their preferences on eight example item topics in this figure. From this figure, we can find that different user segments generated by **EMP-kmeans** imply variant preferences.

TABLE 3
Relative precision gain when training only on *click events* over training on original whole dataset.

Model	$prec_1$	$prec_2$	$prec_3$	$prec_4$	$prec_{10}$
EMP	1.81%	-1.57%	-1.79%	-3.65%	-1.72%
EMP-agegender	16.23%	16.07%	15.41%	13.65%	12.58%
EMP-kmeans	20.54%	22.05%	26.39%	26.50%	22.44%
EMP-tensor	22.86%	24.33%	21.02%	22.20%	19.79%

In this experiment, we also compare the CTR prediction performance between using the original dataset and using the data with only *click events*, i.e. only the users' clicks and their views along with one click. Table 3 presents the relative precision gain of the model trained on dataset with only *click events* over that trained on original whole dataset. From this table, we can find that excluding those *non-click events* and *click-other events* from training set can boost the performance of the CTR prediction especially when used together with segmentation methods. Interestingly, however, there is very little difference for baseline EMP model and it performs even a little worse if only using *click event* for training. We can also observe that excluding *non-click events* and *click-other events* can improve performance further if the user segmentation is more accurate for personalized recommendation, which can further indicate that users with higher engagement in the content module are more essential for learning the recommendation model with higher level of personalization. Since there is no personalization for the baseline EMP, excluding those *non-click events* and *click-other events* from training set can give extremely limited help to performance. In the following experiments, we will conduct more experiments to further demonstrate how user engagement is a critical factor for the effectiveness of personalized recommendation.

5.3 Effects of user engagement

We now explore effects of different types of user engagement (Section 4.2.1) 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 4 demonstrates the relative precision

TABLE 4
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%

gains when training without *click-other events* over training on original whole dataset. This table clearly shows that excluding *click-other events* can improve CTR estimation.

As discussed in Section 4.2.1, besides *click events* and *click-other events*, there are also *non-click events* in which users had no action after they opened the Yahoo! front-page. 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 8 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'

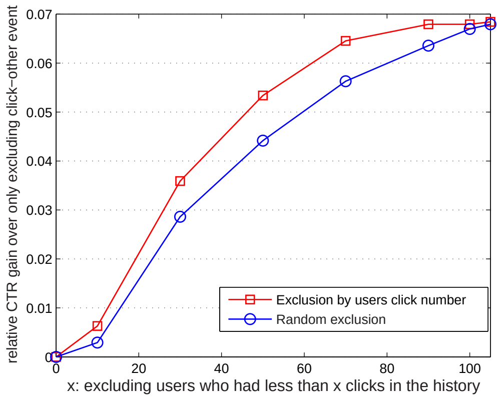

Fig. 8. Relative precision gain when training with data after excluding some *non-click events* over training with data excluding only *click-other events* (using EMP-kmeans).

previous engagement is a good signal to judge the users' potential engagement in future events.

Furthermore, we compare the performance of all three types of user engagement. As shown in Figure 8 and Table 4, exclusion on some *non-click events* can benefit more to the recommendation performance than exclusion on only *click-other events*. After comparing Table 4 with Table 3, we can find that **EMP-kmeans** perform better when training only on *click events* rather than training with excluding *click-other events* and some *non-click events*. We hypothesize that, even after excluding *click-other events* and *non-click events*, there are still a lot of view samples that do not contribute to the recommendation model; but, **EMP-kmeans** trained only on *click events* can remove all of these non-useful view samples so as to improve the recommendation model. However, training only on *click events* may also remove some useful view samples at the same time. The experiments in the next subsection (e.g. Figure 9) will verify our hypothesis.

5.4 Effects of position bias

According to all the above experiments, we have proved that using click event data is the most effective way for online learning based on user segmentation model. Therefore, we use click event data to validate the position bias approach in this section. As discussed in Section 4.2.2, we use relative average CTR values provided in Figure 3 to estimate position weights for learning samples.

Table 5 shows the result by position-weighted approach. From the table, we can find that, although precision_1 is

TABLE 5
Position-weighted result over equal sample result.

Model	$prec_1$	$prec_2$	$prec_3$	$prec_4$	$prec_{10}$
position weighted	2.3%	-0.4%	-0.9%	-0.1%	-0.2%

improved by 2.3% over the baseline where all learning samples are equal weighted, the improvement is not significant especially because precision values at other positions are hurt.

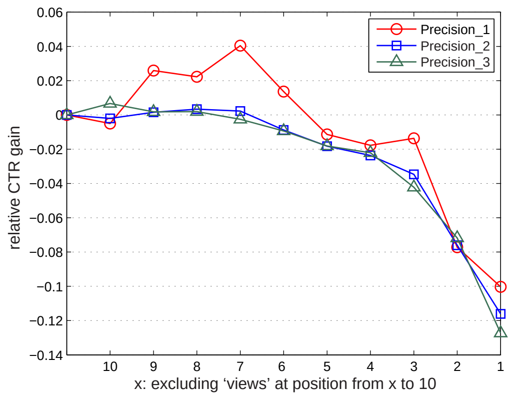

Fig. 9. Relative precision gains by removing view samples at the positions close to bottom.

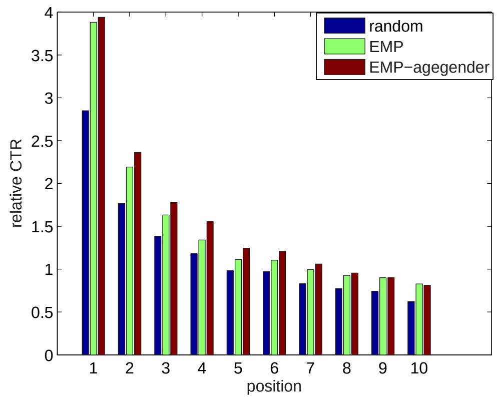

Fig. 10. CTR comparison in bucket test at different positions, where relative CTR is computed by scaling the absolute CTR by a constant factor.

We consider that there are two reasons for this observation. First, since the average CTR value is estimated over different queries and different time, as shown in Equation 5, the CTR variations regarding different queries at different times are quite large. Moreover, as we use *click events* data for learning, it implies that the user must have examined the clicked item in the event. Therefore, the probability that an item located at Position j has been examined by the user (i.e. $P(\text{exam} \mid \text{pos} = j)$ in Equation 3) does not only depend on the prior position weight but also the context of this item in the specific event, such as the items located right at the previous or next position. Hence, the estimation of $P(\text{exam} \mid \text{pos} = j)$ should incorporate the click context in addition to the prior position weight.

We also try another empirical approach based on the intuition that the closer an item is located to the bottom position, the more likely it is ignored by users. Therefore, we remove view samples at a few positions that are close to bottom position. We try removing views at Position 10, Position 9 and 10, Position 8, 9 and 10, and so on, until we remove the view samples at all positions. Figure 9 shows the precision results by these different view sample removal strategies. We observe that removing view samples at Position 10, 9, 8, and 7 yields best precision at position 1. With more positions' view samples removed, the result becomes worse as the quality of view samples is too limited despite the higher quality of view samples.

Comparing the results using the position weighted approach with those using the view sample removing approach, the position weighted approach does not yield significant advantage, which also implies that CTR variations regarding different queries at different times are large.

5.5 Bucket test results

Beyond the simulation results we previously presented, we conduct online-test regarding the proposed models on real users. The online-test is called *bucket test*, in which different models are applied simultaneously to real user visit traffic, but they are all limited in small parts of the whole traffic, respectively. Such *bucket test* enable us to compare performance of different models by observing

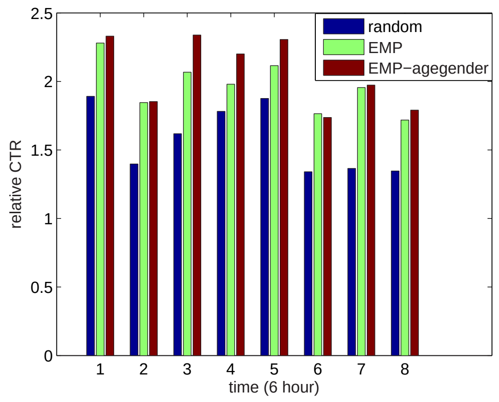

Fig. 11. CTR comparison in bucket test at 8 consecutive 6-hour time slots during 2 days, where relative CTR is computed by scaling the absolute CTR by a constant factor..

their respective recommendation results in different small parts of user visit traffic.

In our experiments, we apply two models in *bucket test*, which are the **EMP** model (baseline, Section 3.2) and the **EMP-agegender** model (Section 4.1.1). Regarding learning samples for model update, **EMP** model uses all events while **EMP-agegender** model uses only click events. By offline results in Table 3, we have observed that the **EMP-agegender** model is better than the **EMP** model. Now we will compare them by *bucket test*. More concretely, we divide the whole user visit traffic into three different buckets. The first bucket is our random learning bucket, which only occupies a small amount of user visit traffic. The other two buckets are serving buckets, which rank trending queries by the **EMP** model and the **EMP-agegender** model respectively, and display the trending queries to the users within the corresponding buckets. These two serving buckets have similar amounts of user visit 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 about 6.01% higher than the **EMP** model bucket.

We also compare CTRs at different positions and different times, as shown in Figure 10 and Figure 11 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 such relative CTRs in different buckets. In Figure 10 and Figure 11, 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 as we discussed in Section 3.2.

In terms of the CTR improvement of the **EMP-agegender** model over the **EMP** model, Figure 10 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 11, 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.

In future, we will also bucket-test other good models including the **EMP-kmeans** and the **EMP-tensor** model.

5.6 Discussions

We have studied a few action-interpretation-based approaches including user segmentation, user engagement and position bias. The success of user segmentation for personalization is due to the fact that the proposed clustering algorithms actually group users by interests and preferences that are implicitly demonstrated by their behaviors. Once the interest patterns are determined by clustering algorithms, a user will be assigned to a segment by her profile features. Fortunately, user profile features also highly correlate with behaviors and interests. Thus, the user segment assignment is usually reliable except when the user is new to the site so that her profile features are poor. Although K-means algorithm and tensor segmentation algorithm yield similar precision performances, the K-means algorithm is much more preferred due to its efficiency. First, it saves human labeling efforts which are expensive and unreliable. Moreover, tensor segmentation learning requires much more computation.

User engagement is another important factor. In our user segmentation model, precise action interpretation is critical for online learning as the samples in each segment are relatively sparse. In the real product, we have tested the user segmentation model using just click events. Similar to the offline results presented in previous sections, the online CTR result is also significantly improved over the non-segmentation model using all events.

Position bias is a sophisticated factor. Using fixed position weights is not very effective to further improve CTR estimation. For a click event, a strong signal is that the clicked item should be more attractive to the user than the rest of non-clicked items. However, for our per-item model approach, the CTR of each item is estimated independently by a Poisson process assumption so that such comparison information in click events are lost. While per-item model is easy for product implementation, we need to further study new models which can utilize such competing preference information.

6 Conclusion

In this paper, we have studied a few important topics towards exploring user action interpretation for online personalized content optimization. We build a online personalized content optimization system using the *parallel-serving-buckets* framework. In this framework, we introduce action interpretation for both more effective user segmentation and better understanding on the informativeness of different user actions. In particular, we leverage users' click actions to group homogeneous users into the same segment; then, we explore the effects of a couple types of user engagement factors as well as the position bias on the online learning procedure. Large-scale evaluations on both offline dataset and online traffic of a commercial portal website demonstrate that we can significantly improve the performance of content optimization by integrating all of these user action interpretation factors into the learning process. In the future, we are interested in exploring more information about personalization, such as users' geographic location and click behaviors from Web search, and studying how to taking advantage of them to benefit content optimization.

References

[1] D. Agarwal, 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.

[2] D. Agarwal, B.-C. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *Proc. of KDD*, 2010.

[3] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *NIPS*, 2008.

[4] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proc. of SIGIR*, 2006.

[5] J. Ahn, P. Brusilovsky, J. Grady, D. He, and S. Y. Syn. Open user profiles for adaptive news systems: help or harm? In *Proc. of the 16th International World Wide Web Conference (WWW)*, 2007.

[6] D. Billsus and M. Pazzani. Adaptive news access. In *The Adaptive Web - Methods and Strategies of Web Personalization*, 2007.

[7] A. Broder. Computational advertising and recommender systems. In *Proc. of the 2nd ACM International Conference on Recommender Systems (RecSys)*, 2008.

[8] G. Buscher, L. van Elst, and A. Dengel. Segmentation-level display time as implicit feedback: a comparison to eye tracking. In *Proc. of SIGIR*, 2009.

[9] W. Chu and Z. Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. In *Proc. of the 12th International Conference on Artificial Intelligence and Statistics*, 2009.

[10] W. Chu and S. T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proc. of the 18th International World Wide Web Conference (WWW)*, 2009.

[11] W. Chu, S. T. Park, T. Beaupre, N. Motgi, and A. Phadke. A case study of behavior-driven conjoint analysis on yahoo! front page today module. In *Proc. of KDD*, 2009.

[12] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. of the 16th International World Wide Web Conference (WWW)*, 2007.
[13] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. of WWW*, 2007.
[14] L. M. de Campos, J. M. Ferndez-Luna, J. F. Huete, and M. A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785 – 799, 2010.
[15] D. Downey, S. Dumais, D. Liebling, and E. Horvitz. Understanding the relationship between searchers' queries and information goals. In *Proc. of CIKM*, 2008.
[16] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proc. of the 13th International World Wide Web Conference (WWW)*, 2004.
[17] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *Proc. of SIGIR*, 2004.
[18] Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. In *Proc. of CHI*, 2007.
[19] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of SIGIR*, 1999.
[20] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of IJCAI*, 1999.
[21] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *Proc. of SIGIR*, 2004.
[22] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. of SIGIR*, 2005.
[23] D. Kelly and N. J. Belkin. Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback. In *Proc. of SIGIR*, 2001.
[24] D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. In *Proc. of SIGIR*, 2004.
[25] H.-N. Kim, I. Ha, K.-S. Lee, G.-S. Jo, and A. El-Saddik. Collaborative user modeling for enhanced content filtering in recommender systems. *Decision Support Systems*, 51(4):772 – 781, 2011.
[26] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. In *Communications of the ACM*, 1997.
[27] D. Lewis. Applying support vector machines to the trec-2001 batch filtering and routing tasks. In *Proc. of TREC*, 2002.
[28] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of the 19th International World Wide Web Conference (WWW)*, 2010.
[29] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proc. of SIGIR*, 2010.
[30] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *Proc. of IUI*, 2010.
[31] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proc. of AAAI*, 2002.
[32] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proc. of the 16th International World Wide Web Conference (WWW)*, 2007.
[33] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proc. of CHI*, 2004.
[34] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12:1247–1283, 2000.
[35] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. of SIGIR*, 2006.
[36] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *Proc. of WWW*, 2007.
[37] Y. Wind. Issue and advances in segmentation research. In *Journal of Marketing Research*, 1978.
[38] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proc. of WWW*, 2012.
[39] Y. Yang, S. Yoo, J. Zhang, and B. Kisiel. Robustness of adaptive filtering methods in a cross-benchmark evaluation. In *Proc. of SIGIR*, 2005.
[40] K. Yu, V. Tresp, and S. Yu. A nonparametric hierarchical bayesian framework for information filter. In *Proc. of SIGIR*, 2004.
[41] Y. Zhang and J. Koren. Efficient bayesian and hierarchical user modeling for recommendation systems. In *Proc. of SIGIR*, 2007.
[42] P. Zigoris and Y. Zhang. Bayesian adaptive user profiling with explicit and implicit feedback. In *Proc. of CIKM*, 2006.

Jiang Bian is a Scientist at Yahoo! Labs. He obtained his PhD degree in Computer Science from Georgia Institute of Technology, Atlanta GA, in 2010. Prior to that, he received the BS degree in Computer Science from Peking University, China, in 2006. His research interests include information retrieval, Web mining, social network analysis, and machine learning. More details of his research and background can be found at https://sites.google.com/site/jiangbianhome/.

Anlei Dong is a Scientist at Yahoo! Labs. He obtained his PhD in electrical engineering from the University of California, Riverside for his work on Perceptual Concept Learning in Image Databases. Previously, he received BS and MS degrees in automation from the University of Science and Technology of China. Anleis current research interests are information retrieval, machine learning and data mining, especially deep dive into the modeling approaches to provide the most relevant contents to Internet users in the applications of both search and recommendation.

Xiaofeng He is a Senior Dev at Microsoft. He obtained his PhD from Pennsylvania State University. Xiaofengs current interests include Web search, query rewriting, learning to rank, and Chinese relevance. Prior to joining Microsoft, Xiaofeng was a Research Scientist at Yahoo! Labs, where his research consisted of keyword suggestion, Ads quality index, vertical search, and content optimization.

Srihari Reddy is a Software Engineer at Google. He have worked broadly in the area of NLP, IR and machine learning. He obtained his MS from Johns Hopkins University where his focus was language modeling for automatic speech recognition. Prior to that, he was a Research Engineer at Yahoo! Labs, where his focus is improving Web search relevance by developing better algorithms, features and data research.

Yi Chang joins Yahoo! in 2006, and he is managing ranking science team in Yahoo! Labs to work on multiple relevance ranking or recommendation projects. His research interests include applied machine learning, information retrieval, natural language processing and text mining. Yi Chang is the author or coauthor of more than 40 referred journal and conference publications.