Probabilistic Models For Ad Viewability Prediction On The Web

Chong Wang, Achir Kalra, Li Zhou, Cristian Borcea, *Member, IEEE* and Yi Chen, *Member, IEEE*

Abstract—Online display advertising has becomes a billion-dollar industry, and it keeps growing. Advertisers attempt to send marketing messages to attract potential customers via graphic banner ads on publishers' webpages. Advertisers are charged for each view of a page that delivers their display ads. However, recent studies have discovered that more than half of the ads are never shown on users' screens due to insufficient scrolling. Thus, advertisers waste a great amount of money on these ads that do not bring any return on investment. Given this situation, the Interactive Advertising Bureau calls for a shift toward charging by viewable impression, i.e., charge for ads that are viewed by users. With this new pricing model, it is helpful to predict the viewability of an ad. This paper proposes two probabilistic latent class models (PLC) that predict the viewability of any given scroll depth for a user-page pair. Using a real-life dataset from a large publisher, the experiments demonstrate that our models outperform comparison systems.

Index Terms—Computational Advertising, Viewability Prediction, User Behavior

1 Introduction

Online display advertising has emerged as one of the most popular forms of advertising. Studies [1] show that display advertising has generated earnings of over $63.2 billion in 2015. Online advertising involves a publisher, who integrates ads into its online content, and an advertiser, who provides ads to be displayed. Display ads can be seen in a wide range of different formats and contain items such as text, images, Flash, video, and audio. In display advertising, an advertiser pays a publisher for space on webpages to display a banner during page views in order to attract visitors that are interested in its products. A *page view* happens each time a webpage is requested by a user and displayed in a browser. One display of an ad in a page view is called an *ad impression*, and it is considered the basic unit of ad delivery.

Advertisers pay for ad impressions with the expectation that their ads will be viewed, clicked on, or converted by users (e.g., the ad results in a purchase). Traditional display ad compensation is mainly based on user clicks and conversion, because they bring direct profits to the advertisers. Much research has been done for predicting click rate and conversion rate [2], bid optimization [3], and auctions [4].

- *Chong Wang is with the Department of Information Systems, New Jersey Institute of Technology, USA.*
  E-mail: cw87@njit.edu
- *Achir Kalra is with Forbes Media LLC and with the Department of Computer Science, New Jersey Institute of Technology, USA.*
  E-mail: akalra@forbes.com
- *Li Zhou is with the School of Information Science and Engineering, Fujian University of Technology, China.*
  E-mail: lideep@foxmail.com
- *Cristian Borcea is with the Department of Computer Sciences, New Jersey Institute of Technology, USA.*
  E-mail: borcea@njit.edu
- *Yi Chen is with Martin Tuchman School of Management, with a joint appointment at the College of Computing Sciences, New Jersey Institute of Technology, USA.*
  This is the corresponding author.
  E-mail: yi.chen@njit.edu

Manuscript received April 03, 2016.

Recently, there is growing interest by advertisers to use online display ads to raise brand awareness and to promote the visibility of companies and their products. Indeed, users like to purchase products from the brands that they recognize and trust. Display ads can create an emotional experience that gets users excited about a brand and builds trust. However, users do not typically click this type of ads, rendering the traditional form of pricing structure based on clicks or conversion to be ineffective.

To address this problem, another pricing model, which pays ads by the number of impressions that a publisher has served, has become popular in the display advertising market. However, a recent study [5] shows that more than half number of the impressions are actually not viewed by users because they do not scroll down a page enough to view the ads. Low viewability leads to ineffective brand promotion.

Therefore, a new pricing model is emerging: pricing ads by the number of impressions that can be *viewed* by a user, instead of just being served [6]. This avoids the frustration of advertisers, who concern about paying for ads that were served but not seen by users.

Not surprisingly, ads placed at different page depths have different likelihoods of being viewed by a user [7]. Therefore, it is important to predict the probability that an ad at a given page depth will be shown on a user's screen, and thus be considered as *viewed*. The vertical page depth that a user scrolls to is defined as the scroll depth.

Viewability prediction is important for many cases:

Guaranteed impression delivery. One of the main ad selling methods is guaranteed delivery, in which advertisers contract publishers to buy guaranteed advertising campaigns. The contracts may fix the number of impressions, targeting criteria, price, etc. As the industry moves toward transacting on viewable impressions, advertisers may propose contracts that specify the number of impressions that will be viewed. Predicting ad viewability helps publishers to fulfill such contracts by placing the ads in the right

impressions.

Real-time impression bidding. Advertisers can also buy impressions through real-time bidding. Given the impression context, including the user, the page, and the ad position, advertisers desire to know the probability that the ad will be in-view. Based on the viewability, advertisers can adjust the bidding price for an impression and improve ad investment effectiveness. Specifically, they can bid higher for impressions with high predicted viewability. In addition, publishers can also benefit from ad viewability prediction by adjusting the minimum prices for impressions which are offered for bidding.

Webpage layout selection. Viewability is expected to become a crucial factor in page layout design, which may impact ad revenue [8]. Publishers are exploring personalized page layouts that can balance ad viewability and user experience. For example, if a user will not scroll deep, the ad slot at the bottom of the page may be moved higher, while considering the impact on user experience.

Recommender Systems. Dwell time (i.e., the time a user spends on a page) has been regarded as a significant indicator of user interest. Recommender systems can also employ scroll depth prediction as another critical metric of user interest.

This paper studies the problem of predicting the probability that a user scrolls to a page depth where an ad may be placed, and thus the ad can be *in-view*. Scroll depth viewability prediction is challenging. First, most users visit only several webpages on a website. It is challenging to detect user interests based on such sparse history of user-page interaction. Second, it is challenging to select significant webpage and user features related to user scrolling. Intuitively, page topics and user interests are regarded as influential factors. But it is non-trivial to explicitly model these features. Naturally, we may resort to latent models that utilize latent features. However, a commonly used latent model, Singular Value Decomposition (SVD), is not suitable to give probabilistic predictions on a full spectrum of scroll depths. Specifically, an SVD model can be trained with data consisting of users, pages, and whether a certain scroll depth is in-view in individual page views, and then be used to predict the viewability for that specific scroll depth. But one SVD has to be trained for each possible scroll depth. Another option is to train an SVD model with data consisting of users, pages, and the maximum page depth that a user scrolls to on a page. The predicted maximum page depth can help give a binary decision for any given scroll depth (i.e., in-view or not), but it cannot give a probabilistic value for a scroll depth to be in-view, which is important to determine pricing. As a webpage typically have multiple ad slots at different page depths (and sometimes ad positions may be even dynamically determined), it may be costly to build one SVD model for every single depth.

We first analyze a real-life dataset collected from a large online publisher and develop a probabilistic latent class model (PLC) with constant memberships (PLC_const) that predicts the viewability of any given scroll depth for a page view. In particular, it learns from training data the user and page *memberships*, i.e., the probability that a user/webpage pair belongs to each latent user/webpage class. The memberships are used to predict the viewability of a page depth. Furthermore, we take webpage features into account and propose another probabilistic latent class model powered by dynamic memberships (PLC_dyn), which can better adapt to changes in user and webpage characteristics, e.g., user interest and webpage attractiveness. "Dynamic" means the final memberships of a user/webpage pair are not directly calculated from training data, but they are determined in real-time based on the feature values. Specifically, compared to PLC_const, PLC_dyn uses two softmax functions powered by linear functions to calculate the final memberships in real-time. PLC_dyn learns from the training data the weights in the linear functions, instead of the final memberships of each user/page pair. The output of the models is the probability that a given page depth is in-view. Compared with a binary decision, i.e. in-view or not, a probabilistic output is very useful in optimization problems, e.g., page layout selection.

The proposed methods have been experimentally compared with four systems: SVD, Cox Regression, Logistic Regression, and a deterministic method. The experiments show that our PLCs have better prediction performance than the comparative systems. Also, PLC_dyn has better adaptability and less memory usage than PLC_const.

Our contributions are summarized as follows: 1) We define the problem of viewability prediction for any page depth. 2) We propose two novel statistical models based on PLC to predict the probability that a page depth will be in-view. 3) We demonstrate experimentally using a real-life dataset that our two PLCs outperform three comparison systems. Compared with PLC_const, PLC_dyn can save more memory and better adapt to changes in user and webpage characteristics.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 4 presents the real-life dataset. Sections 5 and 6 describe the two proposed PLC models. Experimental results and insights are presented in Section 8. The paper concludes in Section 9.

2 Related Work

Researchers have investigated scrolling behavior and viewability for webpage usability evaluation. In [7], [9], [10], the authors discover that users spend more time looking at information on the upper half of the page than the lower half. Also, the distribution of the percentage of content viewed by users follows a Gaussian-like distribution. We differ from these works in our main goal: viewability prediction. Existing work [11], [12] collects scrolling behavior and uses it as an implicit indicator of user interests to measure webpage quality. In contrast, we design algorithms to predict the scrolling behavior for any user-webpage pair.

Several studies have attempted to predict user browsing behavior, including click [2], [13], [14], [15], [16] and dwell time [17], [18]. The existing methods on click prediction are not applicable in our application. They rely heavily on side information (e.g., user profile, and users' queries and tweets) [16] in order to detect what the user is looking for and thereby suggest the items that are more likely to be clicked on. In our application, on the other hand, there is no such kind of explicit indicators of user information needs and detailed user profile. Wang et al. [13] learn user's click

behavior from server logs in order to predict if a user will click an ad shown for the query. The authors use features extracted from the queries to represent the user search intent. In our case, search queries, which can explicitly reflect user interests, are not available.

Most of the work on click prediction [14], [15] is done on the advertiser side, based on high-dimensional features about users (e.g., private profiles), ad campaigns (e.g., ad content), and impression context. However, such data is not accessible at the publisher side. Our goal is to use the publisher data to predict page depth viewability. In addition, Chen et al. [2] propose a factor model to predict if an ad shown together with search results at a specific position will be clicked on. However, this prediction is made for a given position and a query-ad pair, but does not consider the individual users as a factor. In contrast, our methods make predictions that are tailored for individual users and pages. Furthermore, compared with other user responses, scrolling is a more casual behavior because users may terminate the viewing process at any time [19]. In contrast, users do not easily click an item. In other words, clicking is more deliberate, while scrolling is more casual.

For dwell time prediction, Liu et al. [17] fit the dwell time data with Weibull distributions and demonstrate the possibility of predicting webpage dwell time distribution from page-level features. Yi et al. [18] predict dwell time through Support Vector Regression, using the context of the webpage as features. However, both methods do not consider individual user characteristics, an important factor of scrolling prediction.

Our models are also related to meta-level hybrid recommender systems [20], which typically cascade a content-based and a collaborative system. However, existing studies [21], [22], [23] in meta-level category are not applicable in our case. They either require pre-knowledge about user detailed profiles, e.g., gender and age, or pairwise preference of items, which are not available in our case.

In summary, there is no existing research attempt to predict the maximum scroll depth of a user/page pair and to predict ad viewability. In addition, existing methods for user behavior prediction cannot be easily adapted to solve the scrolling behavior prediction problem. Exploring such a problem, our pilot work [24] analyzes a real-life dataset, identifies the features that impacts scrolling behavior, and proposes a probabilistic latent class model that predicts the viewability of any given scroll depth for a page view using constant user and webpage memberships.

3 Problem Definition

Before defining the problem, let us first introduce several important concepts to be used in the problem definition: 1) The *scroll depth* is the percentage of a webpage content vertically scrolled by a user. 2) The *maximum scroll depth* of a page view is how far down the page the user has scrolled during that view. The maximum scroll depth that a user u will scroll on a webpage a is denoted as x_{ua}. 3) The *target scroll depth*, denoted as X, is the page depth whose viewability an advertiser or publisher wants to predict. For instance, a publisher wants to predict the probability that an ad is in-view in a page view. In this case, the target scroll depth can be the percentage of the webpage that contains at least half of the ad. [1]

Our problem is to estimate how likely a user will scroll down to a target scroll depth of a webpage. Specifically, the prediction should be personalized to individual users and webpages. The proposed approach is a supervised learning technique. The inputs of the training module are historical user logs that contain the context of page views. The output is our viewability prediction model. The inputs of the prediction model are a target page depth X and a given pair of user u and webpage a, while the output is the viewability probability of X in the page view.

Problem Definition. *Given a page view, i.e., a user u and a webpage a, the goal is to predict the probability that the max scroll depth, denoted by x_{ua}, is no less than X, i.e., $P(x_{ua} \geq X|u, a)$.*

4 Observations and Analysis of A Real-Life Dataset

We use a proprietary dataset collected over one month on a large publisher's website, i.e., Forbes. The dataset consists of logs of user browsing behavior captured via Javascript events. These scripts send the data to a server. This type of client-side approach accurately captures users' behavior even in multi-tabbed modern browsers [18]. Compared to the dataset used in the pilot work [24], the dataset collected for this article contains more page views and casual users who read more than three web articles on Forbes.com. Since frequent users tend to engage with web articles much more than casual visitors whose behavior is more arbitrary, a flatter max scroll depth distribution is obtained as shown in Figure 2. This makes the prediction much more challenging because user scrolling behavior is relatively more diverse and volatile.

The scroll depth is recorded according to the last row of pixels on users' screens. In this paper, we adopt 1% as the minimum unit of scroll depth; thus, the range of scroll depth is from 0% to 100%. Once a user stops scrolling and stays at a position for one second, the scroll depth is recorded in the user log. Figure 1 shows an example, in which the bottom of the users screen is at the 50% of the whole page. Thus, the scroll depth at the moment is 50%.

Fig. 1: An Example of a Scroll Depth

The user log of this dataset includes user IDs, URLs, user agents, user geo-locations and maximum scroll depths of page views. Individual users are identified by cookies. Table 1 shows some of the important attributes captured in the log. Each row corresponds to a page view. For instance, the max scroll

1. This is in line with the definition suggested by the Interactive Advertising Bureau: a viewable display ad impression requires that a minimum of 50% of pixels be in-view for a minimum of one second. We do not consider the one second in-view duration.

TABLE 1: Example of User Log

User ID	IP	URL	Max Scroll Depth	GMT Time
001	1.3.4.5	/abc	72%	07/12/2015 11:00:00
002	7.6.9.2	/bcd	66%	07/12/2015 11:01:33

depth of the first page view is 72% and that of the second page view is 66%.

The publisher also provides their article metadata which can be retrieved using an API. The article metadata in JSON format contains basic information about each web article, including channels (i.e., topical categories) and sections (i.e., sub-channels). Some of these fields are used in one of the proposed models as webpage features (PLC_dyn).

Figure 2 illustrates the distribution of max scroll depths observed on the publishers' platform. It can also be noticed that there are very few page views whose scroll depths are less than 10%. The main reason is that the top 10% of most webpages can be loaded on the first screen, especially on desktops. In this case, the viewability of the first 10% of webpages is nearly 100%. Therefore, in this research, we mainly focus on the viewability prediction for the page depths greater than 10%.

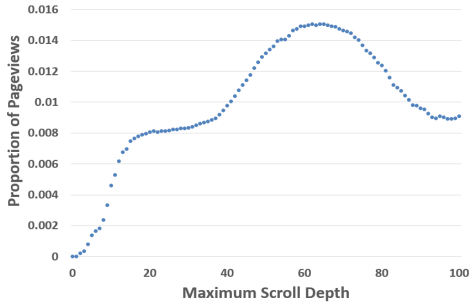
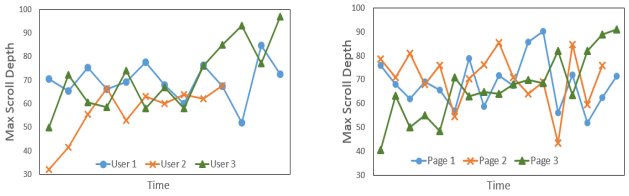


Fig. 2: The Distribution of Max Scroll Depth



(a) Three Random Users (b) Three Random Pages

Fig. 3: The Max Scroll Depths of Three Users and Three Pages across 1-hour Sessions

We investigate how much the max scroll depth varies across sessions. All page views of a user/page occurred in one hour are considered to be in the same session. The max scroll depth of the user/page in a one-hour session is the mean max scroll depths of the page views in the session. Figure 3(a) plots the behavior of three randomly picked users from the users who have more than 10 unique page views in a session. Figure 3(b) plots the same figure for three pages that are randomly selected from the pages which have sufficient visits. The figures indicate that there are significant differences in the user behaviors and in the visits experienced by different pages in terms of max scroll depths. Furthermore, we do not observe any periodical pattern in scrolling behavior, which makes the prediction challenging.

We also extract all page views of each individual user and then calculate the standard deviation of the page views of each user. The standard deviation of each user reflects the variation of the user's reading behavior. We then calculate the mean ($\mu_u = 13.21$) and the standard deviation ($\sigma_u = 9.33$) of the standard deviations of all users. μ_u represents the average variation of user behaviors, while σ_u represents how different the variations are across users. As we can see, the same user behaves quite differently on different pages. We also analyze the behavior difference of different users on the same page, where $\mu_p = 12.50$ and $\sigma_p = 7.87$. This indicates that different users behave differently on the same page. As a reference point, the most challenging case arises when the max scroll depth of any page view is randomly drawn from a uniform distribution, $unif(0, 100)$. In this case, the standard deviation is about 29. Thus, fixing one variable, the mean of the of the group standard deviation μ' is 29 as well. All μ_* are less than μ' because fixing these variables helps control the variation of the outcome to some degree. Meanwhile, most μ_* are more than half of μ', showing that the problem is still challenging.

5 PLC_const: Prediction Model with Constant Memberships

Our task is to infer the max scroll depth of a page view, x_{ua}, where u is the user and a is the webpage. It is intuitive that the characteristics of individual users and webpages can be utilized to improve the performance of max scroll depth prediction models. For example, users who prefer to scroll far down on most webpages would have a higher probability to scroll down the current page. Also, features such as device type and geo-location are easy to be modeled.

However, some other significant features are very hard to capture due to lack of data and the ambiguity of user-webpage interaction. For example, pages with popular content and good design may motivate users to scroll more. But accurately modeling topic popularity and webpage design is difficult. Other examples include user interests and psychology. Therefore, depending solely on explicit features will not lead to accurate prediction.

In addition to feature modeling, data sparsity is another challenge. While a large publisher usually has tens of thousands of webpages, one user only visits several. Likewise, one page may be visited by a small subset of the entire user population. As a result, the user-page interaction employed in prediction could be extremely sparse, which brings about challenges in the prediction performance. A widely-used solution is grouping similar users and similar webpages together and inferring the prediction for a user-page pair using the known data of similar user-page pairs.

To overcome these issues, we use a latent class model [25] to discover classes of users and webpages. Specifically, we build a probabilistic latent class model with constant memberships (PLC_const). The intuition behind it is that different latent classes of webpages and users tend to generate different levels of max scroll depths. PLC_const can detect classes of users and webpages that share similar

patterns of max scroll depth. The exact class memberships of each user and webpage are learnt from the user log and used to do prediction for each page view in test datasets. PLC_const outputs the probability $P(x_{ua}|u,a)$, where x_{ua} is the max scroll depth that a user u reaches on a page a.

Formally, PLC_const works as follow:

$$P(x_{ua}|u,a) = \sum^{N_s}\sum^{N_p} P(s|u)P(p|a)P(x_{ua}|f^{uac},s,p;w_{sp}) \tag{1}$$

where x_{ua} is the max scroll depth of a page view. N_s is the number of latent user classes, and N_p is the number of latent webpage classes. Both N_s and N_p are pre-defined as model parameters. The optimal values for these parameters can be explored by cross validation. $P(s|u)$ is the probability that user u belongs to the latent user class s, while $P(p|a)$ is the probability that webpage a belongs to the latent webpage class p. For simplicity, in this paper, we use s and p to notate individual latent user classes and latent page classes. The last term, $P(x_{ua}|f^{uac},s,p;w_{sp})$, represents the probability that the max scroll depth of the page view is x_{ua}, given the latent user class s and webpage class p. f^{uac} is the feature set that reflects the user, the webpage, and context information, while w_{sp} is the corresponding feature weights.

As mentioned above, the last term can be approximated by the probability density function of a normal distribution. Note that there is no single distribution that can fit all datasets. This paper proposes a general framework for predicting user reading behavior. The proposed methods do not rely on properties specific to the Gaussian distribution. Therefore, different publishers and advertisers can plug in other distributions according to their own datasets. They only need to change the probability density function (Equation 2) and the corresponding M-step.

$$P(x_{ua}|f^{uac},s,p;w_{sp}) = \frac{1}{\sqrt{2\pi\sigma_{sp}^2}}\cdot\exp\left(-\frac{(x_{ua}-w_{sp}^T\cdot f^{uac})^2}{2\sigma_{sp}^2}\right) \tag{2}$$

The right side of Equation 2 is developed based on the probability density function of a normal distribution, i.e., $\frac{1}{\sigma\sqrt{2\pi}}\cdot\exp(-\frac{(x-\mu)^2}{2\sigma^2})$. The mean of the distribution, μ_{ua}, can be modeled by a regression whose features are extracted from the history of u and a as well as the context of the page view, i.e., $\mu_{ua} = w_{sp}^T\cdot f^{uac}$. The superscript uac means the feature set includes user, webpage, and context features. Each pair of latent user class s and latent webpage class p has a set of w_{sp*}, i.e., the weights in the linear function of μ_{ua} and σ_{sp}, i.e., the mean and the standard deviation.

Based on the observations presented so far, we consider seven features:

- User Features:
  1) The mean max scroll depth of all page views of u. This feature captures user browsing habits.
  2) The most recent three max scroll depths of u. This feature captures the recent scroll behavior of the user.
- Webpage Features:
  3) The mean max scroll depth of a by all users. This feature captures the popularity of the webpage.
  4) The most recent three max scroll depths of page views of a. This feature captures the recent scroll behavior for this webpage.
- Interaction of User and Webpage:
  5) Interaction of the mean max scroll depth of u and that of a, i.e., the product of features 1 and 3.
- Page View Context:
  6) User geo-locations, which were shown to be important by our analysis of the dataset.
  7) Device Type (i.e., desktop, mobile, or tablet), also shown to have a certain relevance by our analysis.

Let **W** be the collection of the weight vectors of all latent user classes and webpage classes. $\boldsymbol{\sigma}$ is the collection of the standard deviations of all latent user classes and webpage classes. The features help iteratively determine **W** and $\boldsymbol{\sigma}$.

In Equations 1 and 2, there are several parameters ($P(s|u)$, $P(p|a)$, **W**, $\boldsymbol{\sigma}$). They can be calculated by maximizing the following likelihood function:

$$l(P(s|u),P(p|a),\mathbf{W},\boldsymbol{\sigma}) = \sum_{u,a}\ln\left(\sum^{N_s}\sum^{N_p} P(s|u)P(p|a)P(x_{ua}|f^{uac},s,p;w_{sp})\right) \tag{3}$$

To maximize it, the Expectation Maximization (EM) Algorithm is adopted. The EM algorithm is widely used to solve the maximum-likelihood parameter estimation problem. The EM algorithm performs an expectation step (E-step) and a maximization step (M-step) alternatively. The E-step creates a function for the expectation of Equation 3. This function, i.e., Equation 4, is evaluated using the current estimates of the parameters. The initial values of the parameters are randomly generated.

$$P(s,p|f^{uac};w_{sp}) = P(s|u)P(p|a)\cdot\frac{1}{\sqrt{2\pi\sigma_{sp}^2}}\cdot\exp\left(-\frac{(x_{ua}-w_{sp}^T f^{uac})^2}{2\sigma_{sp}^2}\right) \tag{4}$$

The M-step updates the parameters in Equation 4, which can maximize Equation 3. In each iteration, the M-step updates the value of each parameter based on the result of the E-step. The updated $w_{sp}{}^*$ of each iteration in Equation 7 can be determined by Limited-memory BFGS, an optimization algorithm in the family of quasi-Newton methods.

$$P(s|u)^* \propto \sum_{p,a} P(s,p|f^{uac}) \tag{5}$$

$$P(p|a)^* \propto \sum_{s,u} P(s,p|f^{uac}) \tag{6}$$

$$w_{sp*}^{\cdot} \propto \underset{w_{sp}}{argmax}\{-\sum_{u,a} P(s|u)P(p|a)\cdot[\frac{(x_{ua}-w_{sp}^T f^{uac})^2}{2\sigma_{sp}^2} + \ln\sigma_{sp} + \ln\sqrt{2\pi}]\} \tag{7}$$

$$\sigma^*_{sp} \propto \sqrt{\frac{\sum_{ua} P(s|u)P(p|a)(x_{ua} - w^T_{sp} f^{uac})^2}{\sum_{ua} P(s|u)P(p|a)}} \quad (8)$$

The EM iterations stop if the max ratio is not greater than a pre-defined threshold, which is set to 10^{-3} in our experiments. In other words, it stops if the difference of all feature weights is less than 10^{-3}.

After convergence, the PLC_const with the optimal parameters can predict $P(x_{ua}|u, a)$, i.e., the probability density of any target max scroll depth x_{ua} of a user-webpage pair. Section 7 uses this probability to predict the viewability of any target scroll depth. Similarly, this model can be applied to recommender systems, as mentioned in Section 1. The predicted max scroll depth x_{ua} reflects the interest of the user u in the webpage a.

6 PLC_dyn: Prediction Model with Dynamic Memberships

By computing offline the memberships of users and webpages belonging to latent user and webpage classes, PLC_const predicts the viewability of any target scroll depth in a page view. However, user and webpage memberships in reality can be dynamic during the online process, since user interests and page popularity keep changing. For instance, user interests may shift over time, e.g. from entertainment to sports, which can influence the class memberships of a user. Webpage attractiveness may also change for some reasons, e.g., bursting topics and content freshness. For instance, users viewing a newly updated webpage may scroll deeper than users viewing the same page one week later. The reason is that after one week its content is not fresh and attractive. A drawback of PLC_const is that it can only use fixed memberships calculated from training data to make predictions in test data. For instance, assuming there are two user classes, the memberships of a user in the training data are $s_1 = 0.8$ and $s_2 = 0.2$, i.e., the probability that the user belongs to the first latent user class is 0.8. These memberships are used to predict in all test page views involving that user. Thus, PLC_const cannot adapt user's interest shift.

To capture the dynamic nature of the memberships, we propose to represent the memberships by a function whose output value is determined in real-time. Meanwhile, the feature vectors should also be able to reflect the change of user, webpage, and context. Based on this idea, we develop a dynamic probabilistic latent class model, PLC_dyn that extends PLC_const. This model enables dynamic memberships and also considers webpage information, such as channels, i.e., topical categories (e.g. "finance" and "lifestyle"), and sections, i.e., sub-channels. Webpage information is provided by the article metadata. Note that "dynamic" does not refer to online learning where the model parameters keeps changing based on incoming data stream. The model parameters, i.e. feature weights, are not changed during testing once they have been learnt from the training data. But the memberships calculated based on the model parameters are dynamically changing since feature values may change over time.

Let us clarify the similarities and differences between PLC_const and PLC_dyn in technical details. Similar with PLC_const, PLC_dyn calculates the probability that a user or a page belongs to each class and utilizes user and webpage classes to overcome sparsity. However, unlike PLC_const, PLC_dyn calculates the user and page memberships in real-time, instead of learning constant numbers of memberships from training data offline. In particular, in Equation 1, the memberships $P(s|u)$ and $P(p|a)$ are constant numbers learnt from training data. Before being retrained, PLC_const always uses these fixed memberships to perform predictions for specific users and pages. In contrast, PLC_dyn uses soft-max functions powered by linear functions to calculate user and webpage memberships, as shown in Equation 10. PLC_dyn learns the feature weights in the linear functions from training data, rather than learning final memberships. These feature weights are used to compute the memberships in real-time with the feature values at that moment. Thus, the memberships of a user or a page may be different over time, i.e. dynamic, since the feature values keep updating. For instance, the value of the feature "the mean max scroll depth of the user on the webpages in the same section" is dynamic. It can capture the change of the user's interest. Also, the dynamic value of the feature "the mean max scroll depth of the pages in the same section" can capture the change of topic attractiveness. Thus, PLC_dyn can better adapt to changes of user and page characteristics. To support such calculation, user features and webpage features (webpage features are not used in PLC_const) are used to calculate the user and page memberships, respectively.

Formally, PLC_dyn is modeled as following.

$$P(x_{ua}|u, a) = \sum^{N_s}\sum^{N_p} P(s|f^u; \alpha_s) \cdot P(p|f^a; \beta_p) P(x_{ua}|f^{uac}, s, p; w_{sp}) \quad (9)$$

where $P(s|f^u; \alpha_s)$ represents the probability that the user u with the user features f^u and the corresponding feature weights α_s belongs to the latent user class s, while the $P(p|f^a; \beta_p)$ represents the probability that the webpage a with the webpage features f^a and the weights β_p belongs to the latent webpage class p. $P(x_{ua}|f^{uac}, s, p; w_{sp})$ is the probability that the max scroll depth is x_{ua} given the user and the webpage belonging to s and p respectively. It is almost the same as its counterpart in Equation 1, but they have different feature vectors. f^{uac} is the entire feature set that concatenates all features about the user, the webpage, and the context (e.g. screen sizes, devices), while w_{sp} is the corresponding feature weights. N_s and N_p are the numbers of latent user and webpage classes, respectively.

Equation 9 uses user features f^u and webpage features f^a to calculate the user and webpage memberships, respectively. The parameters that have to be learnt from the training data are feature weights α_s and β_p. In contrast, the memberships in Equation 1 are learnt as constant numbers, $P(s|u)$ and $P(p|a)$. Each user and each webpage receives a set of membership values, which are not subject to change during prediction.

The user membership $P(s|f^u; \alpha_s)$ and the webpage membership $P(p|f^a; \beta_p)$ can be modeled by the soft-max

function [26]. The soft-max function takes the outcome of a linear function as input and outputs the predicted probability for one of the classes given the input vector. User and webpage memberships can be defined:

$$P(s|f^u;\alpha_s) = \frac{1}{Z_u}\exp(\alpha_s^T f^u) = \frac{\exp(\alpha_s^T f^u)}{\sum^{N_s}\exp(\alpha_s^T f^u)} \quad (10)$$

where Z_u is the normalization factor that guarantees the sum of the memberships of a user belonging to all classes is equal to one. The page membership with weights β_p and page features f^a can be modeled similarly. As in PLC_const, the last term can be modeled by Equation 9:

$$P(x_{ua}|f^{uac}, s, p; w_{sp}) = \frac{1}{\sqrt{2\pi\sigma_{sp}^2}}\exp\left(\frac{(x_{ua} - w_{sp}^T f^{uac})^2}{-2\sigma_{sp}^2}\right) \quad (11)$$

f^{uac} is the combination of the user , webpage, and context features. All features are shown as below.

- User Features (f^u):
  1) The mean max scroll depth of the user in past page views, which captures user browsing habits.
  2) The mean scroll depth of the user on the webpages in the same channel.
  3) The availability of the second feature.
  4) The mean scroll depth of the user on the webpages in the same section, i.e., sub-channel.
  5) The availability of the fourth feature.
  6) The mean scroll depth of the users at the same geo location on the webpages in the same channel.
  7) The mean scroll depth of the users at the same geo location on the webpages in the same section.
- Webpage Features (f^a):
  1) The mean max scroll depth of the page by all users. This feature captures the popularity of the webpage.
  2) The mean max scroll depth of the pages in the same channel, i.e., topical category (e.g., finance).
  3) The mean max scroll depth of the pages in the same section, i.e. sub-channel.
  4) The mean max scroll depth of all pages in the "related content list" of the page. If the page has no related content page, it equals to the first feature.
  5) The length of the body text.
- Page View Context (f^c):
  1) Screen Width, i.e., the width of the user's screen.
  2) Screen Height
  3) Viewport Width, i.e., the viewport is the visible area of a web page on user's screen. Unlike screen size, viewport size indicates the area of the user's browser. Viewport size is captured and sent to the server when the user clicks the link of the page.
  4) Viewport Height.
  5) The mean max scroll depth of all page views on the same device.

Note that only the first user feature and the first webpage feature are used in both PLC_const and PLC_dyn. Other features are either new features added in PLC_dyn (e.g., screen size and the mean max scroll depth of the pages in the same channel) or the dynamic version of the features used in PLC_const (e.g., the mean max scroll depth of all page views on the same devices). Also, since the user and webpage characteristics can be reflected in their own memberships, the interaction used in PLC_const is removed.

The new feature set contains many categorical characteristics, e.g., channels, sections, and geo-locations. To reduce the number of dimensions and enable dynamic updates, these categorical characteristics are converted to continuous features. For instance, we convert "device type" (used in the PLC_const) to "the mean max scroll depth of all page views on the same devices" (used in PLC_dyn). Specifically, in PLC_dyn, the continuous variable "the mean max scroll depth of all page views on the same device" is adopted, instead of dummy variables representing devices. This feature in PLC_dyn occupies only one dimension, while its counterpart feature in PLC_const has three dimensions. In addition, the value of PLC_dyns feature is dynamic, since the mean scroll depth is changing over time. In contrast, being represented by dummy variables, the value of PLC_consts feature is constant.

$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\sigma})$ denote the weight vectors of all latent user and webpage classes as well as the weight vectors and standard deviations of all latent user and webpage class pairs, respectively. These parameters can be learnt by maximizing the following likelihood function. Note that the differences between Equations 12 and 3 are the same as those between Equations 9 and 1.

$$l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\sigma}) = \sum_{u,a}\ln\left(\sum^{N_s}\sum^{N_p} P(s|f^u;\alpha_s) P(p|f^a;\beta_p)P(x_{ua}|f^{uac}, s, p; w_{sp})\right) \quad (12)$$

Similar with PLC_const, the EM algorithm is adopted to learn the parameters iteratively in PLC_dyn. The E-step is as below:

$$P(s,p|f^{uac}; w_{sp}) = \frac{P(s|f^u;\alpha_s)P(p|f^a;\beta_p)P(x_{ua}|f^{uac}, s, p; w_{sp})}{\sum^{N_sN_p} P(s|f^u;\alpha_s)P(p|f^a;\beta_p)P(x_{ua}|f^{uac}, s, p; w_{sp})} \quad (13)$$

The values of the parameters are updated in the corresponding M-step using the L-BFGS algorithm:

$$\alpha_{s\cdot}^* \propto \underset{\alpha_{s\cdot}}{argmax}\sum_{u,a}\left[\sum_p P(s,p|f^{ua})\right]\cdot \ln\left[\frac{1}{Z_u}\cdot\exp\left(\alpha_s^T f^u\right)\right] - \frac{\lambda}{2}\alpha_s^2 \quad (14)$$

$$\beta_{p\cdot}^* \propto \underset{\beta_{pj\cdot}}{argmax}\sum_{u,a}\left[\sum_s P(s,p|f^{ua})\right]\cdot \ln\left[\frac{1}{Z_a}\cdot\exp\left(\beta_p^T f^a\right)\right] - \frac{\lambda}{2}\beta_p^2 \quad (15)$$

$$w^*_{sp\cdot} \propto argmax_{w_{sp\cdot}} \sum_{u,a} P(s,p|f^{ua}) \cdot \ln\left[\frac{1}{\sqrt{2\pi\sigma^2_{sp}}} \cdot \exp\left(\frac{\left(x_{ua} - w^T_{sp} f^{uac}\right)^2}{-2\sigma^2_{sp}}\right)\right] \quad (16)$$

$$\sigma^*_{sp} \propto argmax_{\sigma_{sp}} \sum_{u,a} P(s,p|f^{ua}) \cdot \ln\left[\frac{1}{\sqrt{2\pi\sigma^2_{sp}}} \cdot \exp\left(\frac{\left(x_{ua} - w^T_{sp} f^{uac}\right)^2}{-2\sigma^2_{sp}}\right)\right] \quad (17)$$

Note that the first terms of Equation 14 and 15 are not strictly convex. Therefore, adding weight decay, i.e., the second terms, will take care of the numerical problems associated with soft-max regression's over-parametrized representation. The second terms penalize large values of the parameters, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and thus guarantee to have a unique solution, i.e., converge to the global maximum. λ is the weight decay term, which should be greater than 0. In the experiment, it is set to 0.01 based on cross validation. After convergence, the PLC models with the optimal parameters can predict $P(x_{ua}|u,a)$, i.e., the probability density of any target max scroll depth x_{ua} of a user-webpage pair.

7 Viewability Prediction for a Target Scroll Depth

Given a target scroll depth X and a user-webpage pair, the trained PLC models can be used to compute the probability that the max scroll depth will be X, i.e., $P(x_{ua} = X|u,a)$. As stated in the problem definition, our goal is to predict the probability that a given scroll depth will be in view, i.e., $P(x_{ua} \geq X|u,a)$. Therefore, we integrate $P(x_{ua}|u,a)$ from X to 100%, as shown in Equation 18. The result is the probability that the max scroll depth of the page view will be greater or equal to the target scroll depth X. This means the max scroll depth x_{ua} is at a page percentage no less than X. The upper bound of the max scroll depth is 100%, i.e., the page bottom.

$$P(x_{ua} \geq X|u,a) = \int_X^{100\%} P(x_{ua}|u,a)dx_{ua} \quad (18)$$

8 Experimental Evaluation

This section investigates the following questions: 1) Do the proposed PLC models outperform the comparative systems? 2) Does PLC_dyn have better adaptability than PLC_const? 3) How does the training data size influence the performance of the PLC models? 4) Does PLC_dyn require less memory than PLC_const?

8.1 Experimental Dataset

The dataset has been described in Section 4. After random sampling by users, data transformation, and data cleaning, nearly 1 million page views are in the dataset. To avoid bias, the user log is split into three sets of training data and test data, as shown in Table 2. The experimental results are reported by taking the average over the three datasets. On average, there are 80K unique users and 50K unique webpages that generated 200K page views in a 7-day training set and 50K page views in a 1-day test set.

TABLE 2: Training and Test Data Partitioning

#Set	Training Data (7d)	Testing Data (1d)
1	07/06/2015-07/12/2015	07/13/2015
2	07/09/2015-07/15/2015	07/16/2015
3	07/12/2015-07/18/2015	07/19/2015

8.2 Comparison Systems

We compare the performance of the proposed models (PLC_const and PLC_dyn) with several other systems:

Deterministic Method (DET): We compute the proportion of the page views whose max scroll depths are greater or equal to the target scroll depth X in each training set. This proportion is the prediction for all page views given X. For instance, $P(x_{ua} \geq 30\%|u,a)$ is 0.7590 means that the viewability x_{ua} for all test page views is 0.7590.

Logistic Regression (LR): Since one LR model cannot predict for every given target scroll depth, we train an LR model for each target scroll depth. We use the same set of input features as those used to train PLCs. When training a model for a specific target scroll depth, for each page view, we examine whether that page depth was in-view or not. If yes, it is considered to be positive (i.e., viewed), otherwise negative (i.e., not viewed). For instance, when the target scroll depth is set to 20%, a page view is considered as a positive example if its 20% depth was viewed according to the user log. When testing, given the feature vectors of a test page view, the LR model outputs the probability that X is in-view, i.e., $P(x_{ua} \geq X|u,a)$. This probability can be further converted into a binary decision.

Cox Regression (Cox): The research problem can also be considered as a survival analysis problem by treating reaching the max scroll depth as the subsequent event. Thus, we build a Cox regression, commonly used in survival analysis, as a comparison system. Cox regression is defined as $h_k(t) = h_0(t) \cdot \exp(\beta^T x_k)$, where $h_k(t)$ is the probability that a user k does not reach the max scroll depth at depth t. $h_0(t)$ is the baseline or underlying hazard function and corresponds to the probability of reaching the max scroll depth when all the xs are zero. β is the weight vector of the feature set x. The Cox regression is implemented using Lifelines [27], which is a publicly available Python library.

Singular Value Decomposition (SVD): In addition to dimension reduction, SVD is often used to predict a target variable based on historical data. For any $M * N$ matrix A of rank r, SVD can decompose it as $A = U \sum V^T$. U is a $M*M$ orthogonal matrix that spans the "column space". V is a $N * N$ orthogonal matrix that spans the "row space". $\sum$ is a $M * N$ diagonal matrix whose first r entries are the nonzero singular values of A. Using matrix factorization, SVD maps both row items (e.g., users) and column items (e.g., pages) to a joint latent factor space, such that the interactions of row items and column items are modeled as inner products

in that space. In our case, it generates a vector to represent each user or page. The dot product of a user vector and a webpage vector is the prediction of their interaction. Unlike the PLCs, SVD does not utilize the distribution of max scroll depth and the explicit features of page views.

Our SVD model implementation is based on libFM [28]. The number of factors is set to 8, as suggested in the manual. The matrix A is a user-webpage matrix. Each cell value is either 1 or 0, i.e., whether X is in-view or not. The output for a page view is a value between 0 and 1, which is treated as the probability that X is in-view. This probability can be converted into a binary decision. Similar to LR, we build an SVD model for each X.

8.3 Metrics

RMSD: The RMSD measures the differences between the values predicted by a model, $\hat{y}_i$, and the values actually observed, y_i. It is widely used in various research fields and is defined as the square root of the mean square error. If the target scroll depth X is in-view, y_i = 1; otherwise, y_i = 0. $\hat{y}_i$ is the probabilistic prediction of the ith page view, i.e., $\hat{y}_i \in [0, 1]$. The lower RMSD, the better the prediction performance.

Precision, Recall and F1-score: The probability that X is in-view can be converted to 0 or 1, i.e., if it is greater or equal to 0.5, then X is in-view; otherwise, X is not in-view. Thus, the probabilistic prediction problem can be considered as a binary classification problem as well. Hence, precision, recall, and F1-score can be used to compare the models. The precision of a class is the number of page views correctly labelled as belonging to the class divided by the total number of page views labelled as belonging to the class. High precision means high true positive rate and low false positive rate. The recall of a class is the number of page views correctly labelled as belonging to the class divided by the total number of page views that belong to the class. High recall means high true positive rate and low false negative rate. The F1-score of a class is the harmonic mean of the precision and recall of the corresponding class.

Area Under Curve (AUC): The AUC is a common evaluation metric for binary classification problems, which is the area under a receiver operating characteristic (ROC) curve. An ROC curve is a graphical plot that illustrates the performance of a binary classifier system, as its discrimination threshold is varied. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings. If the classifier is good, the true positive rate will increase quickly and the area under the curve will be close to 1. Higher values are better.

8.4 Effect of Parameter Combination

We investigate the performance of PLC_const and PLC_dyn with different combinations of N_s and N_p parameters. As a reminder, N_s is the number of latent user classes, while N_p is the number of latent webpage classes. As one of the ad slots placed at the 60% page depth on a Forbes' article webpage, 60% is used as the target scroll depth X in this experiment for setting the parameters. In fact, this also is one of the most challenging page depths, since it is in the middle of a page. Grid search and random search are adopted in order to find the optimal parameter combination. For the grid search, we try all combinations of $N_s \in [2, 15]$ and $N_p \in [2, 15]$. For the random search, we try 20 combinations of $N_s \in [2, 30]$ and $N_s \in [2, 30]$ which are not included in the grid search. The range of RMSDs is [0.4581, 0.4665] for the PLC_const, while that of the PLC_dyn is [0.4425, 0.4566].

TABLE 3: RMSDs of Different Parameter Pairs

	PLC_const	PLC_dyn
$N_s = 1$, $N_p = 1$	0.4654	0.4501
$N_s = 5$, $N_p = 5$	0.4637	0.4443
$N_s = 6$, $N_p = 8$	0.4609	**0.4425**
$N_s = 8$, $N_p = 7$	**0.4581**	0.4475
$N_s = 10$, $N_p = 10$	0.4601	0.4467

Table 3 shows the 5-fold cross validation RMSD results for different N_s and N_p combinations. For the sake of brevity, we only present partial results, including the best performance. PLC_const and PLC_dyn obtain the best performance with $N_s = 8$ and $N_p = 7$, and $N_s = 6$ and $N_p = 8$, respectively. In the following experiments, we use these values.

8.5 RMSD Comparison

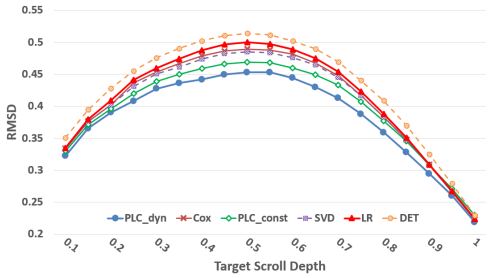


Fig. 4: RMSD Performance

The performance is measured at various target scroll depths by RMSD. Since the top 10% of a webpage are usually shown in the first screen without the user performing any scrolling, we set the range of the target scroll depth to the interval [0.1, 1].

Figure 4 presents the results of RMSD comparison. The results indicate that both PLC_const and PLC_dyn consistently outperform the comparison systems. The percentages of the difference between the PLC_dyn and the PLC_const falls in the range of [2%, 7%] with the mean of 5%.[2]

According to our observation in Forbes user browsing log, the first 20% of webpages are in-view in more than 80% of all page views. Also, the last 10% of webpages are in-view in less than 10% of all page views. Therefore, all models have better performance near the top and bottom of a webpage than in the middle. Their performance near the

2. For each target scroll depth, we calculated what percentage the RMSD of the PLC_dyn is lower than that of PLC_const. We then take the minimum (2%), the maximum (7%), and the mean (5%) of the resultant percentages.

top and bottom is also very similar, which is why the curves overlap in the intervals [0.1, 0.2] and [0.9, 1].

It is increasingly difficult to make correct prediction with the target scroll depth X moving toward the middle of pages. The reason is that the likelihood of being in-view and the likelihood of being not in-view are getting close. The models are more prone to incorrect predictions in the middle of pages. Therefore, RMSDs in the interval (0,2, 0.9) are higher than those in the two tails. Nevertheless, the two proposed PLC models still perform substantially better than the other models within this challenging interval. In the very middle of web pages, i.e., the interval [0.4, 0.6], the deterministic method generates errors that are higher than 0.5. This is because that user browsing behaviors are quite noisy, in which case the overall in-view rates learnt from the training data may not hold very well in the test data. In addition, since it depends only on explicit features and cannot utilize any latent factors, LR does not perform as well as SVD and the two PLC models, which identify latent features or latent classes, respectively. Cox regression has comparable performance with SVD. The curves of these methods almost overlap. Compared to SVD, Cox regression does not make predictions collaboratively; however, it considers multiple auxiliary features to identify the context and the history of users and pages. Cox has lower RMSD than LR because it conditions on the user not leaving the page before the target scroll depth. LR, on the other hand, treats every user-page-depth observation as independent. Matrix factorization-based methods like SVD and Factorization Machines (FM) can handle relatively sparse datasets. However, real-life datasets, such as the one we use, are extremely sparse. For example, when considering only users who read at least three pages and pages read by at least three users, the density of our dataset is less than 0.0006. Even though SVD and FM use matrix factorization to overcome the sparsity issue, they still rely on the sparse interaction of users and pages to infer latent features. In contrast, our models rely on the interaction of classes, instead of that of individual users and pages. Note that we do not aim to solve the cold-start problem. We still expect each page and user to have at a minimal historical browsing history so as to calculate their feature values. In the experiments, the proposed models outperform SVD. Also, although technically these methods could be used in our application, they would have to be re-trained frequently to update according to the changes of user interests and page characteristics, which may introduce additional maintenance overhead or even disruption to business operation. In contrast to PLC_const, PLC_dyn leverages explicit web page metadata, e.g. channels, sections, and related webpages, in order to better identify the latent classes for webpages. It also utilizes more context information, to boost prediction performance. The adaptability provided by dynamic memberships can also contribute on the improvement.

It is difficult to present the effectiveness of all features in our proposed models because there are too many sets of feature weights: Each feature in the PLC_const has $N_s * N_p$ weight vectors, while each user or page feature in the PLC_dyn has $N_s + N_s * N_p$ or $N_p + N_s * N_p$ weights, respectively. Thus, we only investigate the feature weights in the third terms of Equations 1 and 10. The reason is that the first term is user membership and the second term is page membership. The third term directly determines the max scroll depth. Focusing on the best model, i.e., PLC_dyn, we compute the average weight of each feature. The top five significant features in the PLC_dyn are: 1) the mean max scroll depth of the webpage (0.3069), 2) viewport height (-0.1202), 3) the mean max scroll depth of the user (0.1030), 4) the mean max scroll depth of pages with related content (-0.0652), and 5) the mean max scroll depth of pages in the same section (0.0392). All features are already normalized within the range [0, 1]. The p-values of these features are all less than 0.001.

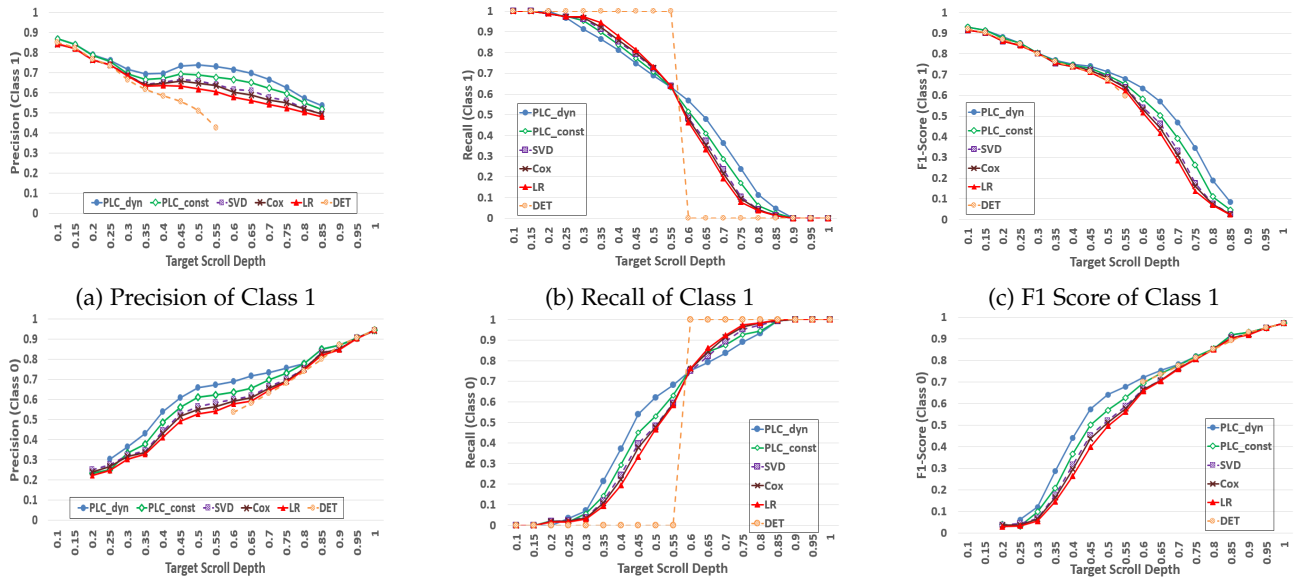
The first three features show that the scrolling behavior in a page view is related to the current viewport size and the historical behavior of the user and the page. Interestingly, the deeper a user scrolled in the pages with related content, the less the user will scroll in the current page. This may be because the user has already been familiar with the content. Thus, the user will probably not read the whole content. The fifth feature indicates that the more interest the user has in the broad topic of the page (i.e., section), the more the user will engage with the page.

8.6 Classification Comparison

False positives (i.e., impressions which are mistakenly considered to be in-view) cause advertisers to invest on ad opportunities that will not be seen by users. This leads to significant investment ineffectiveness. On the other hand, false negatives (i.e., impressions which are mistakenly considered to be not in-view) make publishers lose the revenue that they are supposed to gain because these impressions could have been sold at higher prices. Currently, when bidding an ad opportunity, advertisers consider all ads near the bottom of the page as rarely-seen impressions and thus submit very low bid prices. Thus, identifying both in-view and not in-view impressions is equally important. There are two examples illustrate this goal: 1) because the viewability of page bottoms tend to be low, it is important to recognize in which page views the bottoms will be in-view. 2) Because the viewability of page tops tends to be high, it is important to identify the page views whose tops will not be in-view.

Figure 5 plots the precision, recall, and F1 score of both class 0 and class 1 (i.e., not in-view and in-view, respectively). PLC_dyn overall performs the best among the methods, followed by PLC_const. The performance of class 1 of all methods is high when the target scroll depth X is placed in the interval [0.1, 0.5], since the top of most page views can be in-view. Although it is challenging, the two PLC models can better identify the page views whose tops are not in-view (due to high precisions and recalls of class 0 in the interval [0.1, 0.55]). Similarly, the two PLC models also can better identify the page views whose bottoms are in-view (due to better precisions and recalls of class 1 in the interval [0.6, 1]).

In the interval [0.25, 0.55], both PLC methods have relatively low recall for class 1. The reason is that they classify more page views to class 0 than the comparative systems. A majority of these predictions are correct, i.e., true negatives, while a few are incorrect, i.e., false negatives. The correct ones increase the precision and recall for class 0, but



(a) Precision of Class 1 (b) Recall of Class 1 (c) F1 Score of Class 1

(d) Precision of Class 0 (e) Recall of Class 0 (f) F1 Score of Class 0

Fig. 5: Classification Performance Comparison

the wrong ones decrease the recall for class 1 inevitably, as fewer pages are assigned into class 1. This is also the reason why the two PLC methods precision for class 1 is the highest in the interval [0.25, 0.55]. These observations are more apparent in the interval [0.55, 1]. At the cost of sacrificing the recall for class 0, the PLC models achieve decent performance on the precision for both classes as well as the recall for class 1.

The differences among the models in Figure 5 are not as substantial as those in Figure 4 because RMSD is a more sensitive metric. For instance, given a page view whose target scroll depth X is in-view according to the ground truth, the probabilistic prediction of PLC_dyn is 0.8, while that of LR is 0.6. Both methods have the same evaluation results on the classification metrics because the outputs are greater than 0.5. But their performance can be distinguished when looking at RMSD: The PLC's RMSD is 0.2, while LR's is 0.4. In other words, they do not have any difference in the classification performance, but they do in the RMSD performance.

As shown in Figure 5a, all predictive methods have no precision for class 1 in the interval [0.9, 1] in that no page view in the test data is classified into class 1. Therefore, precision cannot be calculated because the number of page views classified into class 1 is the denominator when prediction is calculated and it is 0 in this case. Due to the same reason, the recall for class 1 is 0 in this interval and no F1-score for class 1 can be computed. A similar observation is obtained for class 0 in the interval [0.1, 0.2], as shown in Figure 5d. The reason that no page view is classified into class 1 within [0.9, 1] is that the distributions of the two classes are very skewed in the interval. Particularly, a large majority of page views are not in-view.

Such imbalanced data precludes statistical methods such as ours to work appropriately [29]. Essentially, the classifiers cannot learn well from the skewed data because the training examples are scarce. To overcome this issue, we have tried simple under/over-sampling. But inevitably, the precision has largely decreased. Therefore, mitigating data imbalance remains a task for future work. Note that the deterministic method (DET) is not impacted by imbalanced data because it always makes the same decision for all test page views given an X. Measured by the classification metrics, it performs as well as the other methods at the two tails, especially in the interval [0.9, 1] because the RMSD of DET is also quite close to other methods as shown in Figure 4. Since DET is much simpler and faster, a practical suggestion on viewability prediction is to use DET to predict the viewability of scroll depths in [0.1, 0.2] and [0.9, 1] intervals, while the PLC models should be employed to predict in the middle of pages.

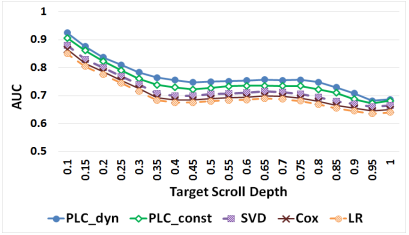


Fig. 6: AUC Comparison

We also use AUC to evaluate the methods. Figure 6 shows that the PLCs outperform other methods. In addition, we notice that AUC decreases from the top to the bottom. To analyze this, we plot the distributions of the prediction outcomes in the positive class and the negative class using box plots. We find that at the top of the page the predictions of both classes are close to 1 due to imbalance in the training data (Class1: median=0.9997, first quartile=0.9993, third quartile=0.9998; Class0: median=0.9972, first quartile=0.9950, third quartile=0.9985). However, the overlap between the prediction distributions of the positive class and the negative class is relatively small: In particular,

at 10%, the first quartile line of the positive class is higher than the third quartile line of the negative class. Therefore, a decision threshold between these two lines can separate the two classes relatively well. In contrast, at the bottom of the page, e.g., 95%, the overlap of the two prediction distributions is more significant (Class1: median=0.0977, first quartile=0.0514, third quartile=0.1714; Class0: median=0.0541, first quartile=0.0305, third quartile=0.0926). The first quartile line of the positive class is much lower than the third quartile of the positive class. Therefore, it is more difficult to separate them by a decision threshold.

8.7 Effect of Latent Classes

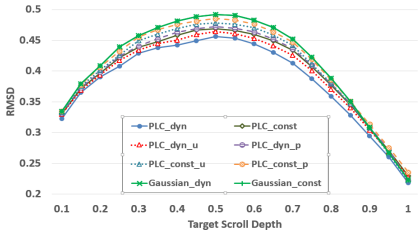


Fig. 7: RMSD Comparison by Considering Only Latent User Classes or Latent Page Classes

Both user groups and page groups are considered in the proposed models. In this section, we evaluate the effects of latent user classes and page classes by separating them out from PLC_const and PLC_dyn one at a time. We also evaluate the performance of PLC models without latent user or page classes. Figure 7 presents the experimental results, in which PLC_const_p and PLC_dyn_p mean the corresponding PLC models with the latent page classes. PLC_const_u and PLC_dyn_u mean the corresponding PLC models with the latent user classes. Gaussian_const and Gaussian_dyn represent the third terms in Equation 1 and Equation 9, respectively.

In both models, PLCs with latent user classes only outperform PLCs with latent page classes. In particular, the RMSD of the PLC_const_p is in fact comparable with SVD. Considering latent user classes in PLC_const instead of latent page classes enhances the performance. A similar observation is also obtained in the PLC_dyn model. This indicates that the reading behavior varies more with the users than with the pages. Although it cannot be denied that pages also play an essential role, the user decisions are the main factors that determine the scrolling behavior.

8.8 Performance on Different Gap lengths

In practice, publishers may not be able to re-train prediction models every day. Thus, it is important to develop models which can adapt to the changes of users and webpages such that the performance stays at a high level for a relatively long time. The goal of this experiment is to evaluate the adaptability of the models. The adaptability is defined as how well a model can adapt to the changes of user factors and/or webpage factors. Such changes in webpage characteristics may influence the class memberships of web pages. This is in fact one of the motivations for using new feature

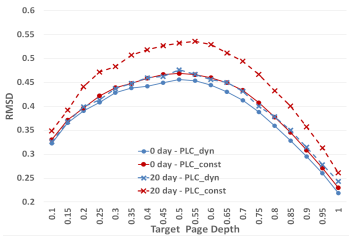


Fig. 9: The RMSDs with 0-day and 20-day Gaps

sets and regression-powered soft-max functions to dynamically compute the memberships of users and webpages in PLC_dyn. Therefore, to test the impact brought by such changes, the two PLC models are compared using a constant value to represent the memberships of users and webpages.

To this end, we re-partition the dataset by varying the time gap between a training set and the corresponding test set. The lengths of the training period and the test period are unchanged, i.e., they are still 7 days and 1 day. But the time period between the training set and the test set, i.e., gap, is varied. For example, setting the gap to 5 days, could use 07/06/2015 - 07/12/2015 as the time period for the training set. The test set is 07/18/2015. Intuitively, the larger the gap, the more likely the user and webpage characteristics are to shift. The gap lengths we adopt are 0 day, 5 days, 10 days, 15 days, and 20 days. Due to the constraints of the time span of the user log, the maximum gap length we set is 20 days.

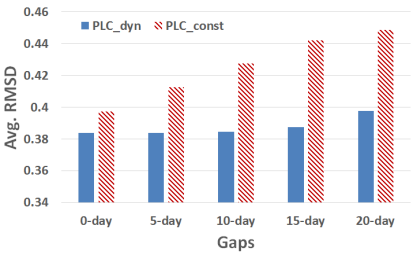


Fig. 8: The Mean RMSDs with Different Gaps across Target Scroll Depths

We only compare the two PLC models because the previous experiments have shown that the comparative systems do not perform as well as the PLC models. Figure 8 shows the average RMSDs at all target scroll depths with different gap lengths. Figure 9 plots the RMSDs of the two models at different target scroll depths with different gaps. To make the curves more distinguishable, we only plot the results of two gaps. The average RMSDs of PLC_dyn are consistently lower than those of PLC_const. The increase of the gap length does not influence significantly the RMSD of PLC_dyn (it stabilizes around 0.384). When the gap reaches 20 days, RMSD increases to 0.3978. The difference between the two models is increasing with the gap because the performance of the PLC_const degrades. This indicates that computing user and webpage memberships in real-time using the soft-max function can adapt well to the dynamic changes of user and webpage factors within the first 15-20 days after the model is trained. In contrast, for PLC_const, the user and web page memberships learnt from the training data cannot remain effective when the gap grows.

According to their requirements, the publishers can decide when the model needs to be re-trained in order to keep high prediction performance upon updating the users and

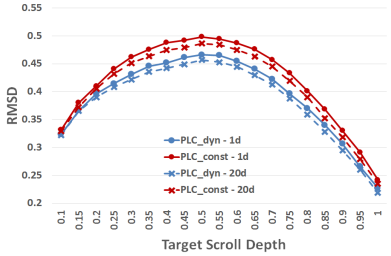
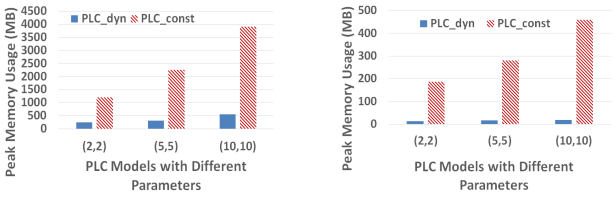


Fig. 11: RMSDs with 1-day and 20-day Training Sizes



(a) Memory Comparison of Training (b) Memory Comparison of Testing

Fig. 12: Memory Usage Comparison

webpage information. For example, a publisher may select 0.5 as the bottom line for RMSD at any target scroll depth. In other words, the model has to be re-trained once RMSD at any target scroll depth increases to 0.5. In this case, based on the experimental results we present, PLC_const needs to be re-trained approximately every 10 days, while PLC_dyn does not have to be updated for more than 20 days.

8.9 Performance on Different Training Data Sizes

TABLE 4: Dataset Partitions with Different Sizes

Training Data	Testing Data (1d)
07/26/2015 (1d)	07/27/2015
07/17/2015-07/26/2015 (10d)	
07/07/2015-07/26/2015 (20d)	

Web sites receive new users and publish new web articles all the time. It is very difficult to draw any inference for these new users and new webpages due to insufficient information about them in training data set. This "cold-start" issue is very prevalent in real-life scenarios. The purpose of this experiment is to test the effect of different training data sizes on the PLC models' performance. Generally, the smaller the training data, the less information is known about users and webpages. The dataset is re-partitioned by fixing the testing dates and varying the time period of the training data, as shown in Table 4. Figure 10 shows the comparison of PLC_dyn and PLC_const in terms of different training data sizes. Figure 11 shows the comparison with 1-day and 20-day at all target scroll depths.

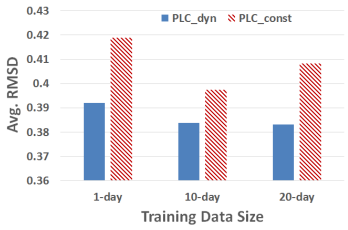


Fig. 10: The Average RMSDs with Different Training Sizes across All Target Scroll Depths

PLC_dyn has better RMSD performance with the increase of the training data size because large training data lead to optimal weight parameters. However, the improvement becomes smaller when the training data size keeps increasing because the optimal feature weights have been obtained. The fact that the PLC_dyn has better performance with small training data indicates that it is more suitable for handling the "cold-start" issue. The performance of PLC_const surprisingly decreases when the training data size increases from 10-days to 20-days. The reason is that the user interest and article attractiveness change over time, which subsequently hurt the prediction performance.

8.10 Memory Usage Comparison

Figure 12 shows the memory usage comparison between the two models. PLC_dyn requires much less memory than PLC_const for both training and testing. The main reason is that PLC_const has to store the memberships of all users and webpages that occur in the training data, which has $N_s \cdot N_{user}$ and $N_p \cdot N_{page}$ memberships. N_s is the number of latent user classes, while N_p is the number of latent webpage classes. N_{user} is the number of users in the training data, while N_{page} is the number of webpages in the training data. In the experiments, N_s is set to 8 and N_p is set to 7. The magnitudes of N_{user} and N_{page} are 10^4. On the other hand, PLC_dyn only has to store the parameters in the linear functions, i.e., $\boldsymbol{\alpha}, \boldsymbol{\beta}$, which have $|f^u|$ and $|f^a|$ numbers, respectively. As stated in Section 6, $|f^u|$ is 7 and $|f^a|$ is 5.

ACKNOWLEDGMENT

This research was supported by the National Science Foundation (NSF) under Grants No. CNS 1409523, DGE 1565478, and CAREER Award IIS-1322406, as well as a Google Research Award, and an endowment from the Leir Charitable Foundations. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

9 CONCLUSIONS

To the best of our knowledge, our research is the first to study the problem of predicting the viewability probability for a given scroll depth and a user/webpage pair. Solving this issue can benefit online advertisers to allow them to invest more effectively in advertising and can benefit publishers to increase their revenue. We presented two PLC models, i.e., PLC with constant memberships and PLC with dynamic memberships, that can predict the viewability for any given scroll depth where an ad may be placed. The experimental results show that both PLC models have substantially better prediction performance than the comparative systems. The PLC with dynamic memberships can better adapt to the shift of user interests and webpage attractiveness and has less memory consumption.

REFERENCES

[1] I. Lunden, "Internet ad spend to reach $121b in 2014," http://techcrunch.com/2014/04/07/internet-ad-spend-to-reach-121b-in-2014-23-of-537b-total-ad-spend-ad-tech-gives-display-a-boost-over-search/.

[2] Y. Chen and T. W. Yan, "Position-normalized click prediction in search advertising," in *KDD'12*, 2012, pp. 795–803.

[3] W. Zhang, S. Yuan, and J. Wang, "Optimal real-time bidding for display advertising," in *ACM SIGKDD'14*, 2014, pp. 1077–1086.

[4] W. Chen, D. He, T.-Y. Liu, T. Qin, Y. Tao, and L. Wang, "Generalized second price auction with probabilistic broad match," in *ACM EC'14*, 2014, pp. 39–56.

[5] Google, "The importance of being seen," http://think.storage.googleapis.com/docs/the-importance-of-being-seen_study.pdf.

[6] M. Mareck, "Is online audience measurement coming of age?" *Research World*, vol. 2015, no. 51, pp. 16–19, 2015.

[7] S. Flosi, G. Fulgoni, and A. Vollman, "if an advertisement runs online and no one sees it, is it still an ad?" *Journal of Advertising Research*, 2013.

[8] H. Cheng, E. Manavoglu, Y. Cui, R. Zhang, and J. Mao, "Dynamic ad layout revenue optimization for display advertising," in *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, 2012, p. 9.

[9] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer, "Not quite the average: An empirical study of web use," *ACM TWEB*, vol. 2, no. 1, p. 5, 2008.

[10] F. Manjoo, "You won't finish this article," *Slate*, 2013.

[11] E. Agichtein, E. Brill, and S. Dumais, "Improving web search ranking by incorporating user behavior information," in *ACM SIGIR'06*, 2006, pp. 19–26.

[12] M. Holub and M. Bielikova, "Estimation of user interest in visited web page," in *WWW'10*, 2010, pp. 1111–1112.

[13] C.-J. Wang and H.-H. Chen, "Learning user behaviors for advertisements click prediction," in *ACM SIGIR'11 Workshop on Internet Advertising*, 2011, pp. 1–6.

[14] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM TIST*, vol. 5, no. 4, p. 61, 2014.

[15] D. Agarwal, B. Long, J. Traupman, D. Xin, and L. Zhang, "Laser: a scalable response prediction platform for online advertising," in *ACM WSDM'14*, 2014, pp. 173–182.

[16] C. Li, Y. Lu, Q. Mei, D. Wang, and S. Pandey, "Click-through prediction for advertising in twitter timeline," in *In Proceedings of KDD'15*. ACM, 2015, pp. 1959–1968.

[17] C. Liu, R. W. White, and S. Dumais, "Understanding web browsing behaviors through weibull analysis of dwell time," in *ACM SIGIR'10*, 2010, pp. 379–386.

[18] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan, "Beyond clicks: dwell time for personalization," in *Recsys'15*, 2014, pp. 113–120.

[19] P. Yin, P. Luo, W.-C. Lee, and M. Wang, "Silence is also evidence: interpreting dwell time for recommendation from psychological perspective," in *KDD'13*. ACM, 2013, pp. 989–997.

[20] F. Ricci, L. Rokach, and B. Shapira, *Introduction to recommender systems handbook*. Springer, 2011.

[21] M. Zanker, "A collaborative constraint-based meta-level recommender," in *In Proceedings of RecSys'08*. ACM, 2008, pp. 139–146.

[22] J. Sun, S. Wang, B. J. Gao, and J. Ma, "Learning to rank for hybrid recommendation," in *CIKM'12*. ACM, 2012, pp. 2239–2242.

[23] X.-L. Zheng, C.-C. Chen, J.-L. Hung, W. He, F.-X. Hong, and Z. Lin, "A hybrid trust-based recommender system for online communities of practice," *IEEE Transactions on Learning Technologies*, vol. 8, no. 4, pp. 345–356, 2015.

[24] C. Wang, A. Kalra, C. Borcea, and Y. Chen, "Viewability prediction for online display ads," in *CIKM'15*. ACM, 2015, pp. 413–422.

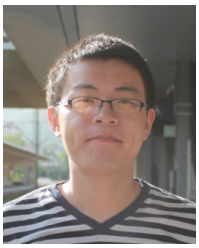
[25] S. Cetintas, D. Chen, and L. Si, "Forecasting user visits for online display advertising," *Information retrieval*, vol. 16, no. 3, pp. 369–390, 2013.

[26] S. Cetintas, L. Si, Y. P. Xin, and R. Tzur, "Probabilistic latent class models for predicting student performance," in *In Proceedings of CIKM'13*. ACM, 2013, pp. 1513–1516.

[27] D.-P. C., "Lifelines," https://github.com/camdavidsonpilon/lifelines, 2016.

[28] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.

[29] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE TKDE*, vol. 21, no. 9, pp. 1263–1284, 2009.



Chong Wang received his Bachelor's degree from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2010. He is currently a Ph.D. candidate in the Department of Information Systems at NJIT. His research interests include machine learning, text mining and computational advertising.

Achir Kalra is a doctoral student of Computer Science at NJIT. He is also a senior vice president of revenue operations & strategic partnerships at Forbes Media. At Forbes Media LLC, he is responsible for programmatic sales, yield management and developing strategic partnerships to grow revenues outside Forbes' traditional display business. His focus is on growing revenues and ensuring that Forbes is at the forefront of a competitive advertising strategy by continuously innovating and creating new products and strategies.

Li Zhou received his Ph.D. degree from Central South University, China, in 2012. He is currently an associate professor in School of Information Science and Technology at Fujian University of Technology, China.

Cristian Borcea is an Associate Professor and the Chair of the Computer Science Department at NJIT, where he has been since receiving the Ph.D. from Rutgers in 2004. He also holds a Visiting Associate Professor appointment at the National Institute of Informatics in Tokyo, Japan. His research interests include: mobile computing & sensing; ad hoc & vehicular networks; and cloud & distributed systems. He has served as the program committee co-chair for Mobile Cloud 2016 and Mobilware 2012. He is a member of ACM and Usenix.

Yi Chen is an associate professor and the Henry J. Leir Chair in Healthcare in the School of Management with a joint appointment in the College of Computing Sciences at New Jersey Institute of Technology (NJIT). Prior to joining NJIT, she was an associate professor in Arizona State University. She received her Ph.D. degree in Computer Science from the University of Pennsylvania in 2005 and B.S. from Central South University in 1999. Her research interests span many aspects of data management. She has served in the organization and program committees for various conferences, including SIGMOD, VLDB, ICDE and WWW, served as an associate editor for DAPD, a guest editor for TKDE and PVLDB, and a general co-chair for SIGMOD'2012.