Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

VIEWABILITY PREDICTION FOR DISPLAY ADVERTISING

by
Chong Wang

As a massive industry, display advertising delivers advertisers' marketing messages to attract customers through graphic banners on webpages. Display advertising is also the most essential revenue source of online publishers. Currently, advertisers are charged by user response or ad serving. However, recent studies show that users barely click or convert display ads. Moreover, about half of the ads are actually never seen by users. In this case, advertisers cannot enhance their brand awareness and increase return on investment. Publishers also lose much revenue. Therefore, the ad pricing standards are shifting to a new model: ad impressions are paid if they are viewable, not just being responded to or served. The Media Ratings Council's standard for a viewable display impression is a minimum of 50% of pixels in view for a minimum of one second. To implement viewable impressions as pricing currency, ad viewability should be accurately predicted. Ad viewability prediction can improve the performance of guaranteed ad delivery, real-time bidding, as well as recommender systems.

This research is the first to address this important problem of ad viewability prediction. Inspired by the standard definition of viewability, this study proposes to solve the problem from two angles: 1) scrolling behavior and 2) dwell time. In the first phase, ad viewability is predicted by estimating the probability that a user will scroll to the page depth where an ad is located in a specific page view. Two novel probabilistic latent class models (PLC) are proposed. The first PLC model computes constant use and page memberships offline, while the second PLC model computes dynamic memberships in real-time. In the second phase, ad viewability is predicted by estimating the probability that the page depth will be in-view for

certain seconds. Machine learning models based on Factorization Machines (FM) and Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) are proposed to predict the viewability of any given page depth in a specific page view. The experiments show that the proposed algorithms significantly outperform the comparison systems.

VIEWABILITY PREDICTION FOR DISPLAY ADVERTISING

by
Chong Wang

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Information Systems

Department of Information Systems

May 2017

APPROVAL PAGE

VIEWABILITY PREDICTION FOR DISPLAY ADVERTISING

Chong Wang

Dr. Yi Chen, Dissertation Advisor — Date
Associate Professor of Management, New Jersey Institute of Technology

---

Dr. Cristian Borcea, Committee Member — Date
Professor of Computer Science, New Jersey Institute of Technology

---

Dr. Vincent Oria, Committee Member — Date
Professor of Computer Science, New Jersey Institute of Technology

---

Dr. Ellen Thomas, Committee Member — Date
Associate Professor of Management, New Jersey Institute of Technology

---

Dr. Nitish Korula, Committee Member — Date
Staff Research Scientist, Google Inc.

BIOGRAPHICAL SKETCH

Author: Chong Wang

Degree: Doctor of Philosophy

Date: May 2017

Undergraduate and Graduate Education:

- Doctor of Philosophy in Information Systems,
  New Jersey Institute of Technology, Newark, NJ, 2017
- Bachelor of Management in Management Information Systems,
  Nanjing University of Posts & Telecommunications, Nanjing, P.R. China, 2009

Major: Information Systems

Presentations and Publications:

Wang, C., Kalra, A, Zhou, L., Borcea, C., Chen, Y. "Predictive Models and Analysis For Webpage Depth-level Dwell Time". *Journal of the Association for Information Science and Technology (JASIST)* (Under Second Round Review)

Wang, C., Kalra, A, Borcea, C., and Chen, Y. "Probabilistic Models For Ad Viewability Prediction On The Web". *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (Under Third Round Review)

Ackerman, B., Wang, C., and Chen, Y. "A Session-Specific Opportunity Cost Model for Rank-Oriented Recommendation". *Journal of the Association for Information Science and Technology (JASIST)* (Under First Round Review)

Zhao, S., Wang, C., Kalra, A, Vaks, L., Borcea, C., and Chen, Y. "Ad Blocking and Counter-Ad Blocking: Analysis of Online Ad Blocker Users". *Proceedings of the 23th Americas Conference on Information Systems.* 2017

Liu, Z., Wang, C., and Chen, Y. "Keyword Search on Temporal Graphs". *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (To Appear)

Wang, C., Kalra, A., Borcea, C., and Chen, Y. "Webpage Depth-level Dwell Time Prediction." *Proceedings of the 25th ACM International Conference on Information and Knowledge Management.* 2016.

Wang, C., Kalra, A., Borcea, C., and Chen, Y. "Revenue-Optimized Webpage Recommendation". *Proceedings of IEEE International Conference on Data Mining Workshop (ICDMW)*. 2015.

Wang, C., Kalra, A., Borcea, C., and Chen, Y. "Viewability Prediction for Online Display Ads". *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. 2015.

Watrous-deVersterre, L., Wang, C. and Song, M., "Concept chaining utilizing meronyms in text characterization". *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*. 2012.

I dedicate my dissertation work to my family and friends. I especially appreciate my parents, Ke Wang and Li An, who raise me up. I also would like to thank my wife, Yaoyu Zhong. She always stands by my side no matter what happens.

Finally, I would like to give special thanks to the eternal light that ignites my bones and guides me to the distance...

ACKNOWLEDGMENT

I would like to thank my advisor Dr. Yi Chen for the guidance and support throughout these years. I would also like to thank Dr. Cristian Borcea, Dr. Vincent Oria, Dr. Ellen Thomas, and Dr. Nitish Korula for taking time to serve on my dissertation committee. I would also like to thank Achir Kalra and my labmates, Shuai Zhao, Jinhe Shi, and Mingda Li, for cooperation. I also would like to thank my family and friends, who always give me support especially when I was at the lowest point of my life.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivation

While traditional advertising may be struggling to prove its effectiveness in the new century, online display advertising is helping to revolutionize marketing. Display advertising provides many benefits that other marketing channels do not, such as faster brand building, effective targeting, and real time conversion measuring. Online display advertising has emerged as one of the most popular forms of advertising. Studies [44] show that display advertising is generating earning of over $63.2 billion in 2015.

Online advertising involves a publisher, who integrates ads into its online web pages, 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. A typical display ad is shown in Figure 1: an advertiser, e.g., Audi, pays an online publisher, e.g., Forbes, for space on webpages to display a banner during page views in order to attract visitors that are interested in the product. Typically, a *page view* happens each time when a particular page on a website is requested by a user and displays in a browser. Also, in display advertising, every occurrence of an ad within the page is called an *ad impression*, which is the basic unit of ad delivery. For instance, one page view is counted when the page in Figure 1 is shown on a user's browser. This page view has one ad impression. The same ad displayed in different page views are considered as different impressions. A large publisher, e.g., Forbes, usually serves billions of impressions in a day.

Advertisers pay for ad impressions with the expectation that their ads can be viewed, clicked on, or even converted by users. One existing display ad compensation

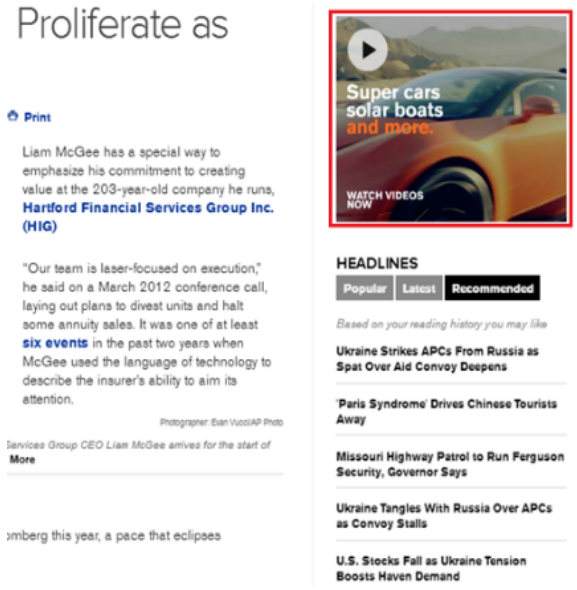


Figure 1.1 An example of a display ad.
Source: https://www.bloomberg.com/news/articles/2013-09-11/laser-focused-ceos-multiply-with-promises-from-ipads-to-macaroni
(accessed on 08/06/2014)

is based on user clicks (pay-by-click) and conversion (pay-by-action). In pay-by-click, an advertiser has to pay for an impression once a user clicks the ad. In pay-by-action, advertisers are charged when the impressions are clicked on or converted (i.e., purchase). These two pricing models bring direct and measurable profits to the advertisers. Thus, they have been widely adopted in sponsored search advertising and display advertising. Much research has been done for predicting click rate and conversion rate [18, 63], bid optimization [89, 16], real-time bidding auctions [15], and user targeting [82, 80]. However, the click and conversion rates are often very low. Users do not typically click this type of ads, rendering the traditional form of pricing structure ineffective. Advertisers cannot achieve their marketing goals and thus lose trust in publishers. Furthermore, pay-by-action is not suitable for certain advertisers, e.g., banks, that do not expect users to immediately purchase their products and service through ads. They just expect users to get familiar with their products and recall them in the future. In addition, click fraud [35] may occur in page-per-click

advertising. An automated script or a computer program imitates a legitimate user of a web browser, clicking on an ad for the purpose of generating a charge per click without having actual interest in the target of the ad's link. This largely hurts the benefits of both publishers and advertisers.

The other display ad compensation is pay-by-impression, in which advertisers pay once an impression is sent to the user side, i.e. *served*. It is highly suitable for advertisers who have growing interest in utilizing online display ads to raise brand awareness and promoting the visibility of the companies and their products. Indeed, users like to purchase products from brands that they recognize and trust. Display ads can create emotional experience that gets users excited about a brand and builds trust. However, a recent study [27] shows that more than half of the ads served were actually not seen by users. One of the main reasons is that users did not scroll down a page enough in order to display an ad, i.e., *in-view*, although technically the ad did load and an impression is served. These invisible impressions are considered to be low-quality in advertisers' eyes because they cannot deliver marketing messages, and thus they hardly enhance return on investment. Thus, low viewability leads to insufficient ad inventory quality and ineffective brand promotion.

Therefore, a new pricing model is emerging: pricing impressions by the number of impressions that can be viewed by a user, instead of just being served [46]. Practically speaking, it means that for brand advertising, advertisers can and will expect guarantees on viewable display impressions. This avoids the frustration of advertisers who are concerned about paying for ads that were served but not seen by users. Thus, it was determined by the Iterative Advertising Bureau (IAB) that the most important need is shifting currency from served impressions to viewable impressions.

Modern online publishers avoid using sticky ads, the positions of which do not scroll with the screen. Although this can almost guarantee 100% viewability, it largely

hurts user experience. Not surprisingly, ads placed at different page depths have different likelihoods of being viewed by a user [26]. Hence, to implement viewable impressions as pricing currency, impression viewability should be accurately predicted. The industry standard of viewable impressions, as developed by the Media Rating Council (MRC), calls for display ads to be viewable if 50% of their pixels are in-view for a minimum of one consecutive second. Therefore, it is important to predict the probability that half of an ad at a given page depth can be in-view for at least one consecutive second.

This study develops machine learning models to predict the viewability of an ad impression placed at any page depth. Ad viewability prediction is important for many applications:

Guaranteed impression delivery. One of the main ad selling methods is guaranteed delivery, in which advertisers contract publishers to buy guaranteed advertising opportunities. It is called "guaranteed" because the contracts may fix the number of impressions, targeting criteria, price, etc. The advertising messages are guaranteed to be served in the page views generating by the targeted audience. The publishers must fulfill the contracts in order to avoid any penalties. 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 (or non-guaranteed delivery). Publishers may also sell remnant ad inventory in real-time when a page view just occurs. The inventory is sold on ad exchanges via real-time bidding. Given from ad exchanges 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. With the ad pricing standard shifting to ad viewability, viewability will become a crucial factor in page layout design, which may impact ad revenue [20]. Simply put, placing all ads at the top of a webpage can increase total viewability. However, this will largely hurt user experience and thus decrease long-term revenue. But placing all ads at the bottom will definitely reduce total viewability. Therefore, 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 may be moved higher, while considering the impact on user experience.

Recommender Systems. User behaviors, such as dwell time (i.e., the time a user spends on a page), have been regarded as significant indicators of user interest. Combining dwell time prediction and scroll depth prediction, viewability prediction can be employed as a critical metric of user interest [75].

This research studies the problem of predicting the viewability of any page depth where an ad may be placed in a page view. Ad viewability prediction tailored for individual page views is challenging. First, most users visit only several webpages on a website. Also, most webpages are visited by only a small number of users. Therefore, it is challenging to detect user interests and webpage characteristics based on such a sparse history of user-page interaction. Second, since ads may be placed at many page depths on a webpage, the viewability prediction model should be able to accept any input page depth. Training a model for each page depth generates 100 models for a page view. In contrast, the proposed prediction model should return the forecasts in real-time by being efficiently trained offline. Third, the optimal viewability prediction should not provide binary output, (in-view or not in-view). Instead, we would ask how

likely it is an ad at 60% page depth will be in-view for at least one consecutive second in a given page view? Such probabilistic output will be very useful in optimization applications. Fourth, to the best of our knowledge, there is no existing work trying to define and predict ad viewability.

1.2 Overview of the Research

This research proposes machine learning models to predict ad viewability in real-time. Note that the proposed methods can be used to predict the viewability of any item on a web page, such as ads, text, and video.

In order to build machine learning models to predict ad viewability in real-time, a real-life dataset is collected from a large online publisher. The datasets record user visiting information, including the geo location of a user, the time that the user read a page, and the behavior that the user performed on the page. A dataset of article metadata is also used in order to get the detailed attributes of page articles.

With the standard definition of viewability suggested by MRC, ad viewability can be predicted from two perspectives:

The first angle is to estimate ad viewability by predicting scrolling behavior. A viewable ad must be shown on a user's screen. Intuitively, whether an ad at a page depth can be in-view is determined by the user's scrolling behavior, i.e., the user must scroll to the page depth where the ad is located before leaving the page. Therefore, the viewability of an ad impression can be estimated by predicting the user's scrolling behavior. In particular, given a page view (a pair of a user and a page), ad viewability can be considered as the probability that the user will scroll to the page depth where the ad is located.

A probabilistic latent class model (PLC) with constant memberships (PLC_const) is developed to predict the viewability of any given page depth for a page view. In particular, from training data, it learns 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, taking into account webpage features, another probabilistic latent class model, powered by dynamic memberships (PLC_dyn), is proposed. PLC_dyn can better adapt to changes in user and webpage characteristics, such as 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, unlike PLC_const, PLC_dyn uses two softmax functions powered by linear functions to calculate the final memberships in the real-time. PLC_dyn learns the weights in the linear functions from the training data, instead of the final memberships of each user/page pair. Both PLC_const and PLC_dyn utilize latent user and webpage classes as well as an observed scroll depth distribution to overcome the data sparsity issue. PLC_const directly learns the probability that a user/webpage belongs to a latent user/webpage class from the training data, while PLC_dyn learns sets of parameters to compute the membership probabilities in real-time. 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 empirical experiments show that the proposed PLC models outperform the comparison systems, including logistic regression, SVD, and Cox regression.

According to the standard definition of viewability, a viewable ad must be shown on screen for at least one consecutive second. This minimum dwell time can also be specified by advertisers based on the types and sizes of their ads. However, predicting ad viewability by scrolling behavior does not take dwell time into account. In addition, not only measuring how long a user stays at a page depth in a page view, the dwell time of the page depth can reflect whether the user scrolls to the page depth as well. If a user does not scroll to a page depth, the dwell time of the page depth must be

zero. The second stage proposes to predict ad viewability by predicting how likely it is that a user will stay at a page depth for at least a certain amount of seconds. This minimum dwell time threshold can be specified by advertisers and publishers.

Thus, in the second phase of this research, a machine learning model is first proposed to predict the viewability of a page depth where an ad is placed. The proposed method can also be applied to predict the dwell time of any items on a page. Specifically, a Factorization Machines (FM) model is adopted because it is able to capture the interaction between input features, overcome the data sparsity issue, and provide flexibility to add auxiliary information. The FM models consider the basic factors (i.e., user, page, and page depth) and auxiliary information. It is determined through experiments that viewport (i.e., the visible area of a user browser), channel (i.e., the topic of the entire article), and Doc2Vec vector (which models the content in the viewport) are the most important auxiliary features. The FM model is evaluated using real-life data from a large web publisher. The experimental results demonstrate that the FM model outperforms deterministic and regression-based comparison models.

In order to discover and leverage the deep patterns among input variables, three deep sequential neural networks are then proposed to predict how likely it is that a given page depth will be viewed by a user for at least a certain dwell time. Ad viewability prediction is considered a sequential labeling problem. The proposed models utilize the information of the previous page depths to predict the viewability of the current page depth. In particular, the models leverage Recurrent Neural Network (RNN) using the Long Short-Term Memory (LSTM) to model sequential dependency into predicting webpage depth viewability. In the first proposed model, LSTM_noInteract, every time step outputs one prediction outcome: the viewability of a page depth. The input of each time step in the proposed LSTM RNN contains information about the user, the page, the depth, and the context. Since user behavior

is determined by the interaction of user, page and depth, the second proposed model considers the interactions by multiplying their embedding vectors before sending the information to the LSTM layers. Furthermore, users often scroll-back on pages. The time a user spent at lower page depths also may indicate the time the user will spend at upper page depths. In addition, in single directional LSTM, predictions made at upper page depths rely on few previous page depths. The third model, Bi-LSTM_Interact, upgrades LSTMs to bi-directional LSTMs, which can take future information, i.e., lower page depths, into account. The experimental results demonstrate that our models outperform the comparison models. The model with the best performance is Bi-LSTM_Interact, which is powered by bi-directional LSTMs and considers embedding interaction.

The contribution of this research includes: 1) We are the first to define and study the problem of viewability prediction, which is a significant problem in online advertising. 2) We are also the first to predict user scrolling behavior in specific page views. 3) We advance the state-of-the-art method to predict depth-level dwell time by developing machines learning models. 4) The proposed models are evaluated using real-life datasets. The experimental results show that our models significantly outperform the comparison systems.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 provides a review of the literature related to this study. Chapter 3 proposes solutions which predict ad viewability by scrolling behavior. Two probabilistic latent class models are developed to conduct such prediction. Chapter 4 presents the proposed methods which predict ad viewability by dwell time. Machine learning models are developed based on factorization machines and deep sequential neural networks. Chapter 5 concludes the thesis and introduces the future research plan.

CHAPTER 2

LITERATURE REVIEW

This chapter introduces the research landscape of online display advertising and the related work on scrolling behavior analysis and dwell time prediction.

2.1 Computational Display Advertising

In display advertising, advertisers pay for showing graphical banners that express their advertising messages in order to boost profits. There are two types of advertising by objective: direct marketing and brand advertising. In direct advertising, through showing graphical ads to website visitors, advertisers expect direct response from potential customers, such as purchase, subscription, voting, and so on. In brand advertising, advertisers create distinct favorable images about their products or services. These display ads that are charged per impression can be used to improve brand awareness. Advertisers have two main approaches to deliver online ads to customers [85]: Guaranteed Delivery and Non-Guaranteed Delivery. In guaranteed delivery, advertisers can directly approach publishers to display ads on their webpages, and if accepted, the publisher will formalize a contract to guarantee the delivery of the requested number of impressions to specific targeting audience. Alternatively, in non-guaranteed delivery, by utilizing the service of an ad exchange, an advertiser can bid ad impressions through real-time bidding (RTB) and has the opportunity to have the ads displayed on the websites of many publishers.

Real-Time Bidding (RTB) is an important aspect of programmatic buying, which is getting more and more popular in display advertising. A publisher sends a bid request of an impression to an ad exchange via the supply side platform (SSP), then to demand side platforms (DSP) to reach advertisers. The impression is sold via an auction. If the advertiser wins the impression, her ad will be displayed to users. To bid

ad impressions, DSPs or advertisers usually determine their own bid prices based on predicted click-through rate (CTR). Extensive research on CTR prediction has been done. The proposed models include regression-based [47, 92, 28, 61], tree-based [49], and neural networks models [23]. In addition, another most important research issue on the advertiser side is the design of effective bidding algorithms. As a proxy of advertisers in RTB markets,a DSP is faced with the task of selecting the appropriate ad impressions and determining their optimal bid prices under budget constraints, aiming at maximizing the ad performance (e.g., the number of ad impressions, clicks or conversions) [88]. Different bidding strategies have been proposed to optimize advertisers' investment [25, 89, 6, 90, 56].

On the other side of the table, display advertising is the most important revenue source. Publishers monetize the visit volume by selling ad impressions of page views. Publishers can sell most of their ad opportunities in advance by guaranteed delivery in which publishers and advertisers formalize advertising contracts. Publishers guarantee to deliver ad campaign message to targeting visitors. The remnant ad inventory is sold through ad exchanges [62] by RTB. Every time, when a user clicks a page link, a page view is triggered. Publishers need to determine which advertising channel (e.g., online channels of RTB, ad networks, and offline channels of contract negotiation) each ad impression on the page goes to. Traditionally, premium ad inventory is always sold via ad networks or offline negotiations, while remnant inventory is left for RTB markets. With the effectiveness of RTB advertising is widely recognized by practitioners, publishers are more inclined to sell premium ad impressions via RTB platforms. As such, how to predict the ad prices and allocate ad impressions among multiple channels accordingly has been intensively studied with the aim to maximize the revenue of publishers [5, 72, 19, 50]. In addition, publishers need to set the reserve price for each impression and submit it to Ad exchanges. The reserve price is the lowest price for publishers to sell the impression. Generally

speaking, a high reserve price may increase the risk that the impression cannot be sold, while a low reserve price may decrease publishers' revenue. As such, researchers have proposed algorithms for optimizing ad reserve prices [53, 86, 58, 51, 38]. Also, different ad sizes and positions have diverse advertising effect. Large ads tend to be viewed, clicked, and even recalled by users. Likewise, ads at the top of pages are more likely to be seen. Therefore, ad format and webpage layout influence bid prices submitted by advertisers. The existing work [20, 69] has focused on selecting optimal ad format for an impression and webpage layout for a page view in real-time.

Existing research has been done on both publisher and advertiser sides. One example loosely related to the proposed research is behavioral targeting. Behavioral targeting comprises a range of techniques used by online publishers and advertisers aimed at increasing the effectiveness of advertising using user web-browsing behavior information [14]. Advertisers can buy user behavior data from third party companies. Although behavioral targeting tracks and analyzes user browsing behaviors, existing research [17, 43, 54, 7, 3] mainly focuses on user-ad interaction (i.e., ad click or conversion). These studies identify user characteristics to perform targeting based on the ads that users clicked or converted in the past and/or search queries that users submitted. However, none of the existing research in behavioral targeting performs user targeting based on implicit user behaviors, e.g., scrolling and dwell time. In contrast, the proposed research attempts to utilize implicit user behaviors to predict ad viewability.

As introduced in the Chapter 1, current pricing model is based on pay-by-serving or pay-by-click/action. Since very few impressions are clicked or converted and only half of the impressions are view by users, the Media Rating Council (MRC) has urged the entire industry to shift the paradigm towards pay-by-view and evaluate advertising campaigns by user engagement, e.g., viewability. In this case, it will be significant to develop a method to accurately predict the viewability of impressions

in real-time. Such research can help advertisers determine the bid prices in RTB and also help publishers to decide the reserve prices of impressions, personalize webpage layout, and so on. Currently, there is no any related research has been done. The proposed research is the first trying to address ad viewability prediction problem. The research activity proposed in this proposal, ad viewability prediction, can be performed on both publisher and advertiser sides. Although the datasets used in this proposal are collected from a large publishers, advertisers can buy similar data from the third party and use the same methods to predict ad viewability as well.

According to IAB, an ad is viewable if half of the ad is in-view for at least one consecutive second. Therefore, viewability prediction can be divided into two parts: 1) predicting the probability that a user will scroll to the page depth where 50% of an ad so that the ad is in-view, i.e., in-view prediction; 2) predicting the probability that the dwell time of the page depth is at least a certain time period given the page depth will be in-view, i.e., dwell time prediction.

2.2 Scrolling Behavior Analysis

Researchers have investigated scrolling behavior and viewability for webpage usability evaluation. In [78, 45, 26], 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 normal distribution. The work in [2, 33] collects scrolling behavior and considers scrolling behavior as an implicit indicator of user interests in order to measure the quality of webpage design and content. However, in contrast to these *analytic* models of scrolling behaviors studied in the past, this proposed research develops a *predictive* model of the scrolling behavior for any user on any page based on historic information. The focus is to make prediction before the user behavior occurs, rather than observation or measurement.

Several studies have attempted to predict other types of user browsing behavior, including click [18, 73, 13, 1] and dwell time (which will be discussed later). For click prediction, one important application is sponsored search, i.e., ads are selected based on user queries submitted to the search engine and shown along with the search results. Chen et al. [18] 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 user as a factor. In contrast, the proposed research makes predictions that are tailored for individual users and pages. Wang et al. [73] 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 the case of the research in this proposal, search queries, which can explicitly reflect user interest, are not available. Most of the existing work on click prediction [13, 1] 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. On the other hand, such data is not accessible at the publisher side. Therefore, these existing techniques of predicting click behavior cannot be readily used to predicting scrolling behavior at the publisher side, which is the goal of this proposed study.

2.3 Dwell Time Prediction

Existing work models dwell time of a whole page as a Weibull distribution [42, 83] or as Gamma distribution [36]. The authors use explicit features, e.g., page length, context, and topics, to estimate the overall dwell time that a user will spend on the whole page. In particular, Liu et al. investigate the feasibility of predicting from page-level features the Weibull distribution of the time that a user spends on a whole webpage. They use Multiple Additive Regression Trees (MART). The features include the frequencies of HTML tags, webpage keywords, page size, the number of

secondary URLs, and so on. They find that page-level dwell time is highly related to webpage length and topics. Yi et al. [83] view the average dwell time of a webpage as one of the item's inherent characteristic, which provides important average user engagement information on how much time the user will spend on this item. The authors present a machine learning method to predict dwell time of article stories using simple features. The features they consider are content length, topical category of the article (e.g., politics, finance, or science), and the context in which the article would be shown (e.g., desktop, tablet or mobile). The authors use Support Vector Regression (SVR) models to predict page-level dwell time. Kim et al. [36] present regression method to estimate the parameters of the Gamma distributions of click dwell time (i.e., the time that the user spends on a clicked result). The features they adopt are similar to those used in Liu et al. In contrast, this proposed research will predict dwell time at a specific depth in a page, which is still an open question. Working at a finer granularity, depth-level dwell time prediction is more challenging than page-level dwell time prediction. Yin et al. [84] run analysis of real data collected from a joke sharing mobile application. The authors find that the dwell time may satisfy a log-Gaussian distribution. They claim that viewing item is such a casual behavior that people may terminate the viewing process at any time. The dwell time varies a lot due to the factors from both items and persons: 1) Items may differ not only in their form and volume (e.g., different length of articles, etc.); 2) They are many subjective human factors to affect the dwell time. For example, different people receive information at different speed and the time of consuming the same item (e.g., reading an article) may differ from person to person. The authors develop a View-Voting model, which can estimate how much a user likes the viewed item according to the item-level dwell time.

In addition to statistical methods, Xu et al. [81] propose a personalized webpage re-ranking algorithm through exploring a user's dwell times in his/her

previous readings over individual documents. According to the cognitive neuroscience phenomenon of semantic satiation, the authors assume that human brain has a fatigue mechanism where the more times a stimulus is repeatedly received by our brain in a short span of time, the less aroused our brain becomes. Then, the authors infer concept word level user dwell times in order to understand a user's personal interest. According to the estimated concept word level user dwell times, the authors can estimate a user's potential dwell time over a new document. Although the proposed algorithm technically can predict the dwell time of any given part of a web page, it assumes that users always read documents carefully so that semantic satiation occurs. However, the proposed algorithm may not be applicable in our application, where users probably do not have patient to read every part of webpages.

In summary, there is no existing research attempt to predict the scrolling behavior or dwell time of a user and webpage pair and to predict ad viewability. In addition, existing methods for user behavior prediction cannot be easily adopted to solve ad viewability problem.

CHAPTER 3

SCROLLING BEHAVIOR PREDICTION

In the first phase, ad viewability is estimated by predicting scrolling behavior. In particular, the goal is to predict how likely it is that a user will scroll to the target page depth so that an ad shown at the page depth will be shown on the screen. This chapter describes the formal definition of viewability prediction by scrolling, the proposed approaches, and evaluation.

3.1 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

[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.

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 1. *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., $P(x_{ua} \geq X|u, a)$.*

3.2 Probabilistic Latent Class Model with Constant Memberships

3.2.1 The Real-Life Dataset

A proprietary dataset is collected over one and a half months on a large publisher's website. It contains more than 1.2 million page views and 100 thousand unique users. 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 [83].

The scroll depth is recorded according to the last row of pixels on users' screens. 1% is adopted 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 3.1 shows an example, in which the bottom of the user screen is at the 50% of the whole page. Thus, the scroll depth at the moment is 50%.

The user log of this project includes user IDs, URLs, user agents, user geo-locations and maximum scroll depths of page views. Individual users are identified by cookies. Table 3.1 shows some of the important attributes captured in the log. Each row corresponds to a page view. For instance, the max scroll depth of the first page view is 72% and that of the second page view is 66%.

Figure 3.2 illustrates the distribution of max scroll depths in our user log. It is observed that the distribution of the max scroll depth generally follows a Gaussian-like distribution. It can also be noticed that there are very few page views whose scroll

Figure 3.1 An example of a scroll depth.
Adapted from https://www.bloomberg.com/photo/oil-extends-drop-below-50-as-u-s-stockpiles-seen-swelling-glut-/-ivpCrFJ0gjPw.html //(accessed on 08/06/2014)

depths are less than 10%. The main reason is that the 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 almost always 1. Therefore, the focus of this research is the viewability prediction for the page depths greater than 10%.

3.2.2 Features Impacting the Max Scroll Depth

The dataset is analyzed to understand which log attributes significantly influence the scroll depth, with the aim of selecting these attributes as features in the prediction model.

Scroll Depth vs. Device Type The reason that page percentage is adopted, rather than pixels, as a measure of scroll depth is because it provides a relative

Table 3.1 Example of User Log

User ID	IP	URL	Max Scroll Depth	GMT Time
001	1.3.4.5	/abc	72%	11/23/2014 11:00:00
002	7.6.9.2	/bcd	66%	11/23/2014 11:01:33

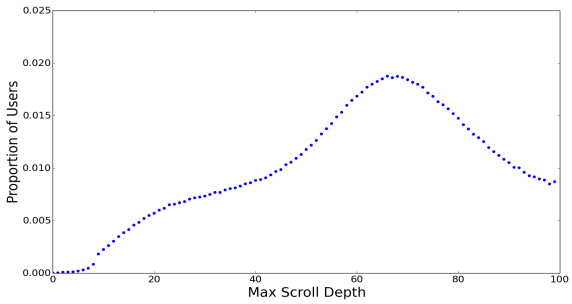


Figure 3.2 Distribution of max scroll depth.

measure independent of device types (i.e., different devices have different screen sizes). If a user reads 50% of a page on a mobile device, while another user reads 50% of the same page on a desktop, it can be assumed that they read the same content of the page. However, this does not deny a hypothesis that devices may affect user behavior which may further influence the max scroll depth. For instance, when reading on mobile phones, users may not have enough patience and may leave the page with little scrolling.

Figure 3.3 illustrates the distribution of the max scroll depth across multiple devices, i.e., desktop, mobile/phone, and tablet. The device type is detected from the user agent attribute. The average max scroll depth is highest on the tablets (65.7%), followed by desktops (61.6%), and mobiles (60.2%). The possible reasons for the overall similar results across devices are: 1) The publisher's webpages are displayed in a mobile-friendly manner; 2) Flicking fingers on the screen is as easy as scrolling the wheel of a mouse [39]. Finally, it is noticed that mobiles, as expected, have certain

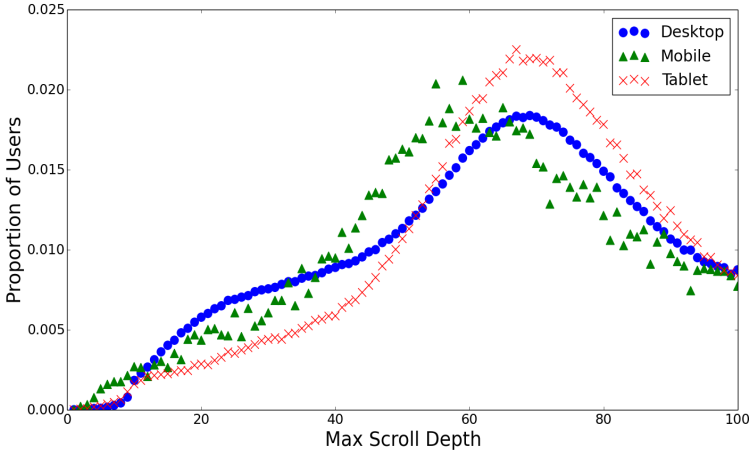


Figure 3.3 Distribution of max scroll depth across devices.

page views with max scroll depth under 15%. This is very rare for desktops. The reasons for such low percentages are: 1) some pages are very long on the mobiles; 2) users close the browser tabs with loaded pages before they view these pages or stop loading the pages before they are shown, in which case the max scroll depth is zero. Although generally similar, the results exhibit a number of differences, and thus the device type is considered as a feature in the proposed model.

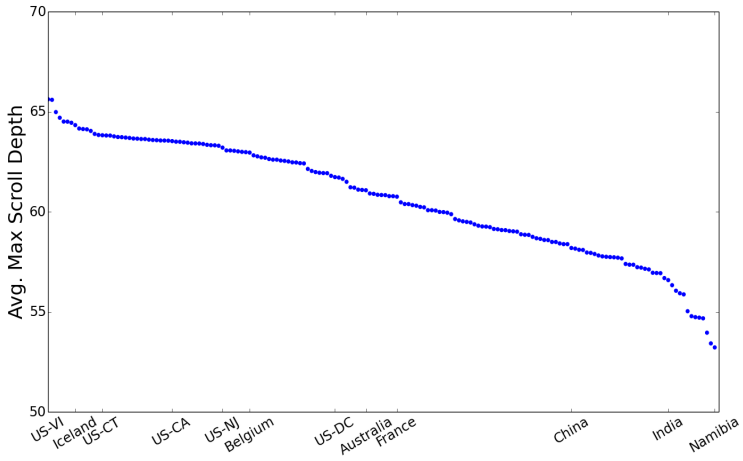


Figure 3.4 Average max scroll depth as a function of user geo-location.

Scroll Depth vs. Geo-location The user log records the countries from which the visitors connect and the US states if the visitors are from US. The locations with sample page view sizes less than 1000 are filtered out. Figure 3.4 shows that most of the top 50 locations for max scroll depth are US states. Interestingly, visitors from U.S. Virgin Islands (65.62%) view pages the deepest, followed by New York State (65.60%) and Texas (65.49%). On the other hand, users from Namibia read the least (53.23%). In addition to user interests and reading habits, user geo-locations may also determine the connection speed, the distance from the publishers' host servers, etc. These factors, independent of users and webpages, may directly play a role on how users engage with the content. Since user geo-location is a significant factor, it is considered as a feature in the proposed prediction model.

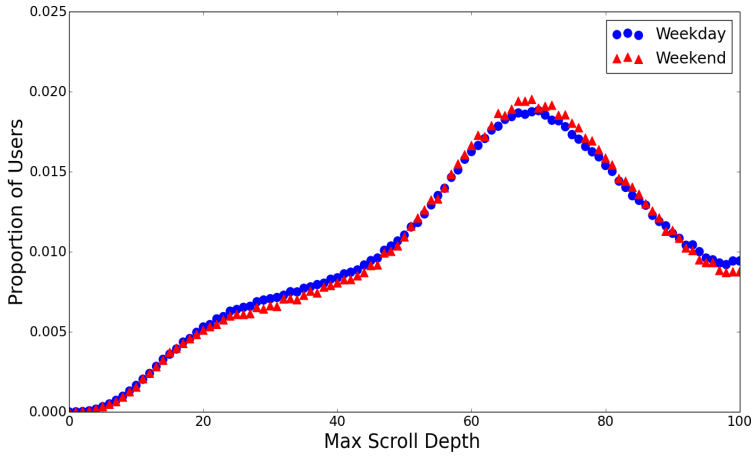


Figure 3.5 Distribution of max scroll depth for week days.

Scroll Depth vs. Day of the Week The day of the week and hour of the day are calculated using the local time of the user which is inferred from the user's IPs and the GMT time in the user log. Figure 3.5 shows that the day of the week does not have a significant impact on the scroll depth. This result contradicts past research [87] which revealed that the day of week determines the impression volume. Thus, this feature is not considered in the prediction model.

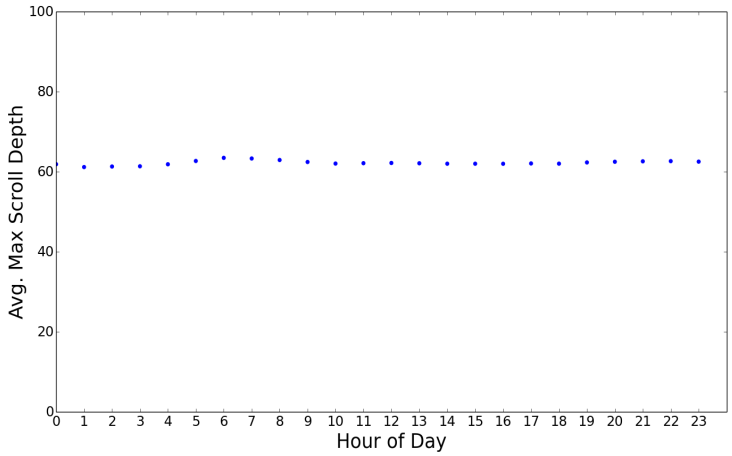


Figure 3.6 Distribution of max scroll depth of different hours of the day.

Scroll Depth vs. Hour of the Day One plausible hypothesis is that users may scroll deepest in the evening, after work. However, surprisingly, Figure 3.6 demonstrates that users seemingly perform very similar at different hours of the day. Thus, the hour of the day is not a significant factor to predict max scroll depth.

3.2.3 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 [11] 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:

$$P(x_{ua}|u,a) = \sum^{N_s}\sum^{N_p} P(s|u)P(p|a)P(x_{ua}|f^{uac},s,p;w_{sp}) \tag{3.1}$$

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 predefined 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 3.2) and the corresponding M-step.

$$\begin{aligned} &P(x_{ua}|f^{uac}, s, p; w_{sp}) \\ &\quad = \frac{1}{\sqrt{2\pi\sigma_{sp}^2}} \cdot \exp\left(-\frac{(x_{ua} - w_{sp}^T \cdot f^{uac})^2}{2\sigma_{sp}^2}\right) \end{aligned} \tag{3.2}$$

The right side of Equation 3.2 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 $\mathbf{W}$ 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 $\mathbf{W}$ and $\boldsymbol{\sigma}$.

In Equations 3.1 and 3.2, there are several parameters ($P(s|u)$, $P(p|a)$, $\mathbf{W}$, $\boldsymbol{\sigma}$). They can be calculated by maximizing the following likelihood function:

$$l(P(s|u), P(p|a), \mathbf{W}, \boldsymbol{\sigma}) = \sum_{u,a} \ln \left(\sum^{N_s} \sum^{N_p} P(s|u) P(p|a) P(x_{ua} | f^{uac}, s, p; w_{sp}) \right) \tag{3.3}$$

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 3.3. This function, i.e., Equation 3.4, is evaluated using

the current estimates of the parameters. The initial values of the parameters are randomly generated.

$$P(s,p|f^{uac};w_{sp}) = P(s|u)P(p|a)\cdot\frac{1}{\sqrt{2\pi\sigma_{sp}^2}}\cdot \exp\left(-\frac{(x_{ua}-w_{sp}^T f^{uac})^2}{2\sigma_{sp}^2}\right) \tag{3.4}$$

The M-step updates the parameters in Equation 3.4, which can maximize Equation 3.3. 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 3.7 can be determined by Limited-memory BFGS, an optimization algorithm in the family of quasi-Newton methods.

$$P(s|u)^* \propto \sum_{p,a} P(s,p|f^{uac}) \tag{3.5}$$

$$P(p|a)^* \propto \sum_{s,u} P(s,p|f^{uac}) \tag{3.6}$$

$$\dot{w_{sp*}} \propto \underset{w_{sp}}{argmax}\{-\sum_{u,a} P(s|u)P(p|a)\cdot [\frac{(x_{ua}-w_{sp}^T f^{uac})^2}{2\sigma_{sp}^2} + \ln\sigma_{sp} + \ln\sqrt{2\pi}]\} \tag{3.7}$$

$$\sigma_{sp}^* \propto \sqrt{\frac{\sum_{ua} P(s|u)P(p|a)(x_{ua}-w_{sp}^T f^{uac})^2}{\sum_{ua} P(s|u)P(p|a)}} \tag{3.8}$$

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 3.2.4 uses this probability to predict the viewability of any target scroll depth.

3.2.4 Viewability Prediction for a Target Scroll Depth

Given a target scroll depth X and a user-webpage pair, the trained PLC_const 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, the 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, $P(x_{ua}|u,a)$ is integrated from X to 100%, as shown in Equation 3.9. 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.

$$P(x_{ua} \geq X|u,a) = \int_{X}^{100\%} P(x_{ua}|u,a)dx_{ua} \tag{3.9}$$

3.2.5 Evaluation

Experiment Datasets To evaluate the proposed method, Forbes' user browsing log is adopted. The user log is split into three sets of training and testing data, as shown in Table 3.2. This was done to avoid bias. The experimental results are reported by taking the average over the three sets. On average, there are 31K+ unique users who generated 300K+ page views in a 10 days training set and 23K+ page views in a 2 days testing set.

Table 3.2 Training and Test Data Partitioning

Set#	Training Data (10d)	Testing Data (2d)
1	11/01/2014-11/10/2014	11/11/2014-11/12/2014
2	11/13/2014-11/22/2014	11/23/2014-11/24/2014
3	11/25/2014-12/04/2014	12/05/2014-12/6/2014

Comparison Systems The performance of the proposed model is compared with three other system described below: a deterministic method, a logistic regression (LR) system, and a singular value decomposition (SVD) system.

Deterministic Method (DET): The proportion of the page views whose max scroll depths are greater or equal to the target scroll depth X is calculated for each training set. This proportion is the prediction for all page views given X. For instance, $P(x_{ua} \geq 30\%|u, a)$ is 0.8953 means that the viewability x_{ua} for all test page views is 0.8953. Formally:

$$P(x_{ua} \geq X|u, a) = \frac{\#pageviews\ whose\ x_{ua} \geq X}{\#pageviews}$$

Logistic Regression (LR): An LR model is built based on the Stanford NLP API. Since one LR model cannot predict for every given target scroll depth, we train an LR model for each target scroll depth. The same set of input features are used as those used to train PLC. The target variable is 1 or 0, i.e., if a page scroll x_{ua} is not less than X, then target variable is 1; otherwise it is 0. When testing, given the features vector 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.

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 PLC, 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 [60]. 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.

Metrics The main metrics we adopt are the Root-Mean-Square Deviation (RMSD) and the F1-score of class 0 (i.e., given scroll depth not in-view) and class 1 (i.e., given scroll depth in-view). We also compare the methods using the precision and recall 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:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$$

where N is the number of test page views. y_i is the ground truth of the ith page view. 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., $\hat{y}_i \in [0, 1]$. RMSD serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of the predictive power of a method. Thus, the lower RMSD is, 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 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.

Effect of Parameter Combination The performance of PLC_const with different combinations of the two parameters, N_s and N_p, shown in Equation 3.1, is investigated. N_s is the number of latent user classes, while N_p is the number of latent webpage classes. Since there is an ad slot located at the 60% page depth on the real webpages analyzed, 60% is taken as the target scroll depth X. Grid search and random search are adopted to find the optimal parameters. For the grid search, all combinations of $N_s \in [2, 12]$ and $N_p \in [2, 12]$ are explored. For the random search, 20 combinations of $N_s \in [2, 30]$ and $N_s \in [2, 30]$ which are not included in the grid search are tried. The range of obtained RMSDs is [0.3637, 0.3683].

Table 3.3 RMSDs of Different Parameter Pairs

RMSD	N_p=4	N_p=5	N_p=6	N_p=7	N_p=8	N_p=9
N_s=4	0.3681	0.3672	0.3678	0.3678	0.3676	0.3659
N_s=5	0.3671	**0.3691**	0.3678	0.3686	0.3675	0.3663
N_s=6	0.3679	0.3676	0.3678	0.3679	0.3671	0.3659
N_s=7	0.3674	0.3679	0.3672	0.3672	0.3645	0.3656
N_s=8	0.3675	0.3678	0.3663	0.3640	0.3672	0.3660
N_s=9	0.3678	0.3671	0.3652	0.3652	0.3638	0.3663
N_s=10	0.3671	0.3673	0.3649	0.3639	0.3644	0.3646
N_s=11	0.3657	0.3644	0.3637	**0.3631**	0.3638	0.3643
N_s=12	0.3640	0.3637	0.3634	0.3636	0.3645	0.3644

Table 3.3 shows the 5-fold cross validation RMSD results for different N_s and N_p combinations. For the sake of brevity, only partial results which contain the best and the worst performance are presented. It is observed that different combinations do not largely influence the performance, with the difference between the best and the worst results being only 0.006. Most parameter combinations generate similar values for precision, recall, and F1-score, respectively.

RMSD Comparison The goal of this experiment is to test the performance of the models with different target scroll depths. Since generally the top 10% of a page can be 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 3.7 plots the RMSD comparison for the four systems. The results show that PLC_const significantly outperforms the three comparison systems. The RMSD performance of the PL_const at all Xs is averagely 10%, and 17% at maximum, better than the second best system, SVD. All models have better performance near the top

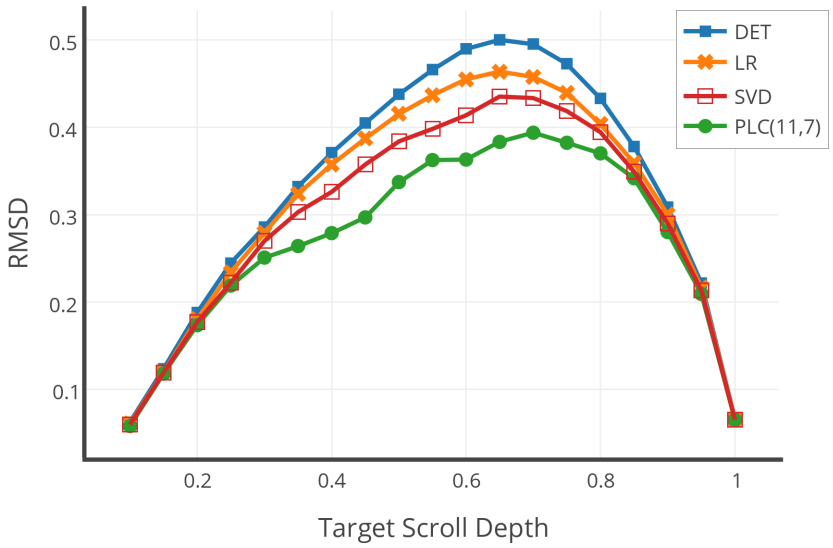


Figure 3.7 RMSD performance.

and bottom of a webpage than in the middle. The reasons for the top of the pages is that most pages are in-view at scroll depths such as [0.1, 0.2]. Being trained by such skewed data, most probabilistic outputs of the models are closer to 0 than 1. Although they may commit mistakes on the cases that are not in-view, the average RMSDs are still relatively low.

The prediction becomes harder with X moving toward the middle of the pages. Intuitively, the models are more prone to making incorrect predictions. Thus, RMSDs in this interval are higher than those in the two tails. Nevertheless, PLC_const performs substantially better than the other systems within this challenging interval. Due to the difficulty of capturing all the significant features, logistic regression does not perform as well as SVD and PLC, which identify latent features or latent classes, respectively.

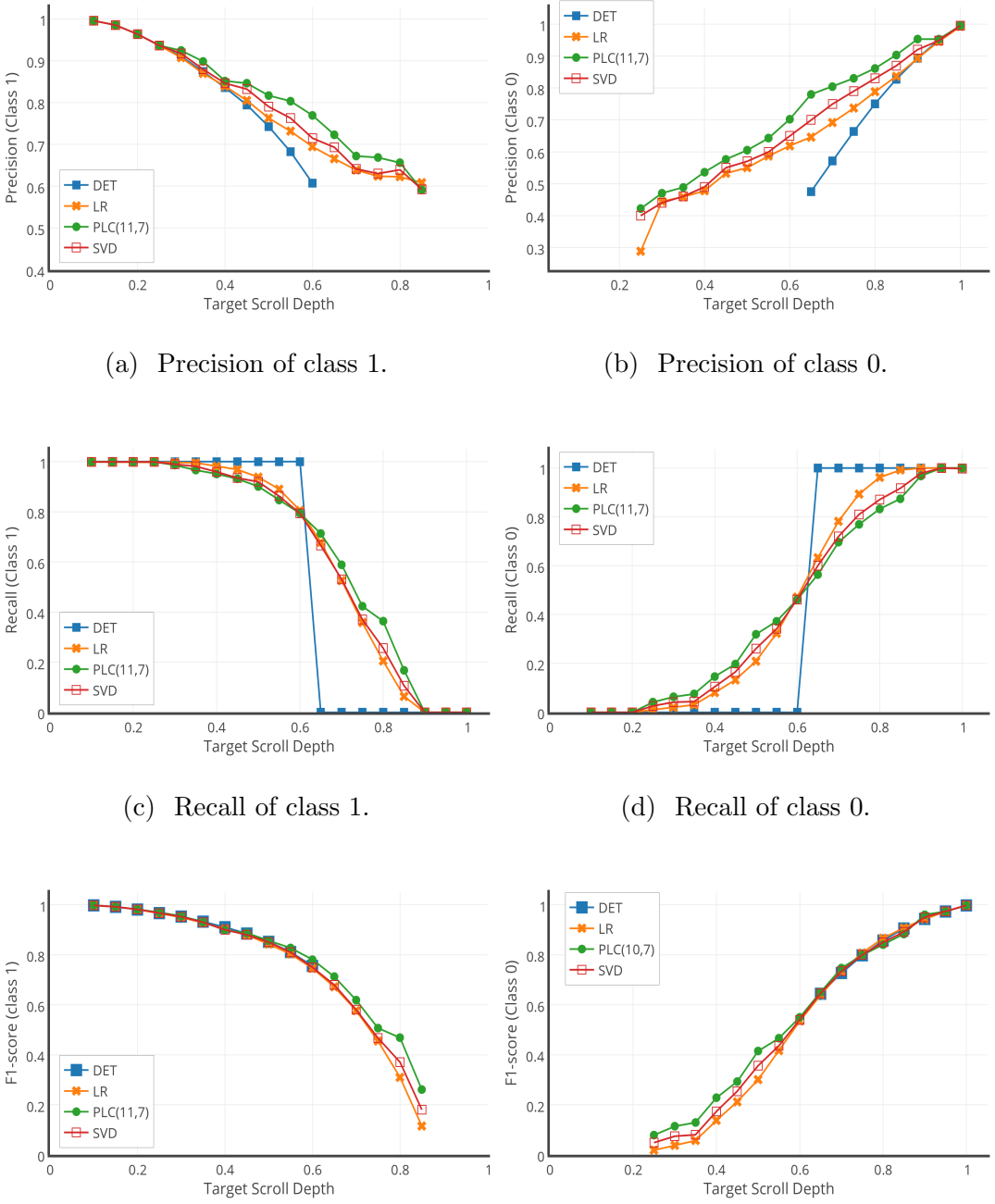
Although RMSD reflects the deviation between probabilistic prediction and ground truth, it cannot tell the whole story of the performance. For example, let us assume there are 100 page views. Given a certain X, the ground truth tells that 99 belong to the not-in-view class and one belongs to the in-view class. A naive model

makes the same prediction, which is 0, all the time. Thus, RMSD for this naive model at X is 0.1, which looks decent. However, such a good RMSD hides the inability of the model to recognize in-view instances. To overcome this issue, we adopt precision, recall, and F1-score to further evaluate our model.

Precision, Recall, and F1-score Comparison Avoiding both false positives and false negatives can improve investment effectiveness for advertisers and increase the ad revenue for publishers. Therefore, identifying both in-view and not in-view impressions is equally important. Two practical examples illustrate this goal: (1) since the viewability of the page bottoms tends to be low, it is important to recognize when the page bottoms are actually in-view; (2) relatively high viewability of the page tops leads to expectations that ads at top are always in-view; however, this is not always the case, and it is very helpful to identify those pages whose tops are not in-view.

Figure 3.8 shows the precision, recall, and F1 score of both class 0 and 1 (i.e., not in-view and in-view). Overall, PLC_const performs the best among the four systems. The performance for class 1 is high when X is set in the interval [0.1, 0.6] because the top of most pages are in-view. Although it is more challenging to recognize the page views whose top is not in-view, PLC_const classifies these page views the best because its precision and recall for class 0 in the interval [0.1, 0.6] are the highest. Likewise, although it is difficult to detect the page views whose bottoms are in-view, PLC_const has the highest precision and recall for class 1 within [0.6, 1].

PLC_const has relatively low recall for class 1 in the interval [0.3 0.6] because it tends to boldly classify more page views to class 0 than the other systems. Most of these predictions are correct, (i.e., true negatives), while just a few are wrong (i.e., false negatives). The correct predictions increase the precision and recall for class 0, but the wrong predictions inevitably decrease the recall for class 1 since fewer page



(a) Precision of class 1.

(b) Precision of class 0.

(c) Recall of class 1.

(d) Recall of class 0.

(e) F1 score of class 1.

(f) F1 score of class 0.

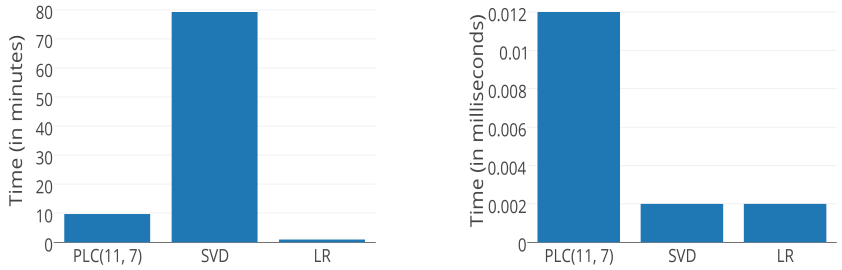
Figure 3.8 Classification performance comparison.

views are classified into class 1. This also explains why PLC_const's precision for class 1 is the highest in the interval [0.3, 0.6]. In the interval [0.6, 1], these observations are even more apparent. At the cost of sacrificing the recall for class 0, PLC_const achieves decent performance on the precision for both classes as well as the recall for class 1.

The differences among the models in Figure 3.8 are not as substantial as those in Figure 3.7 because RMSD is a more sensitive metric. For instance, given a page view whose X is in-view according to the ground truth, the probabilistic prediction of PLC_const 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: PLC_const's RMSD is 0.2, while LR's is 0.4.

LR, SVD, and PLC_const do not have precision results for class 1 in the interval [0.9, 1] because no page view is classified into class 1. Thus, a precision value cannot be calculated because the number of page views labeled in class 1 acts as the denominator in the precision formula and is 0 in this case. For the same reason, the recall for class 1 is 0 in this interval and no F1-score for class 1 can be computed for this interval. A similar behavior happens for class 0 in the interval [0.1, 0.2].

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 like ours to work appropriately [32]. 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.



(a) Avg. training time for entire training set.

(b) Avg. testing time for one test page view.

Figure 3.9 Runtime comparison.

Note that DET is not impacted by imbalanced data because it always makes the same decision for all test page views given an X. It works as well as the other methods in the interval [0.1, 0.2] and [0.9, 1]. 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 PLC_const should be employed to predict in [0.2, 0.8] interval.

Runtime Comparison Figure 3.9 shows the runtime comparison for PLC, LR, and SVD. In this experiment, one PLC_const model is built to predict the viewability of all target scroll depths from 10% to 100%. (step = 5%, so 19 scroll depths). However, for LR and SVD, we build 19 models (the step is 5% for the interval 10% to 100%). Therefore, the time for PLC_const includes one training, while the time for LR and SVD is the sum of 19 trainings. We do not include DET because it does not involve training and makes consistent predictions for all page views for a given X (i.e.. its training and testing runtime are almost 0).

The results show that the training time of LR is much lower than those of PLC_const and SVD because LR does not have to learn any latent patterns from data. Intuitively, learning and applying more latent user classes and webpage classes

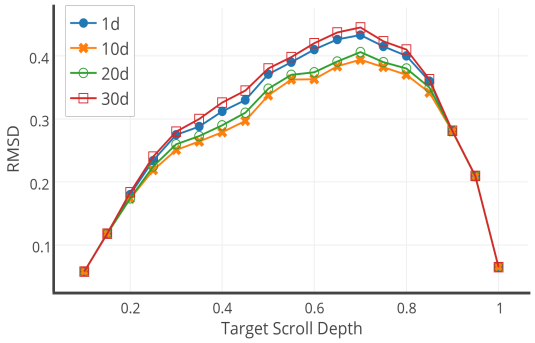
takes more time. Since PLC_const performs better in terms of prediction accuracy, its training time is reasonable, especially compared to SVD. Let us also note that training can be done offline.

The results also show that PLC_const needs more time to make a prediction. However, the absolute value is very low (i.e., 0.012 ms). As an exchange-sold ad is often sold in 200 milliseconds, PLC_const's prediction time can easily be afforded for real-time predictions of incoming pages.

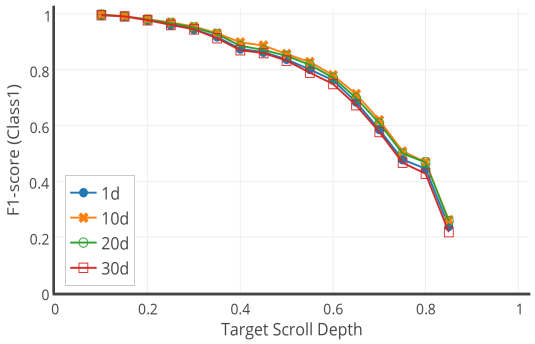
Table 3.4 Dataset Partitions with Different Sizes

Training Data	Testing Data (2d)
11/10/2014 (1d)	11/11/2014-11/12/2014
11/01/2014-11/10/2014 (10d)	
10/22/2014-11/10/2014 (20d)	
10/12/2014-11/10/2014 (30d)	

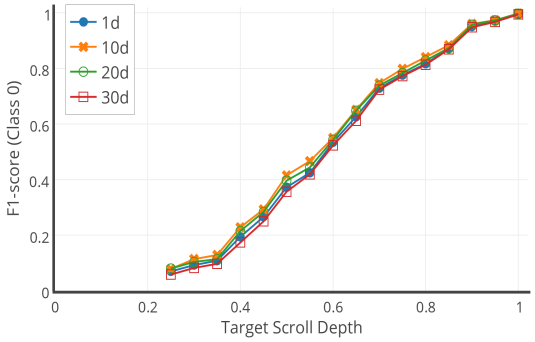
PLC_const Performance on Different Training Data Sizes To test the impact of different training data sizes on the PLC_const's performance, the dataset is re-partitioned by fixing the testing dates and varying the training data sizes, as shown in Table 3.4. All models share the common parameter pair, $N_s = 11$ and $N_p = 7$. According to Figure 3.10, PLC_const results are almost the same for F1 scores. However, the results are distinguishable for RMSD, as this is a more sensitive metric. RMSD for PLC_const(30d) is slightly worse than the others. A possible reason is that the user interest may change over a longer period of time and subsequently hurts the prediction performance. The performance of PLC_const(1d) is not as good as those of PLC_const(10d) and PLC_const(20d) because it utilizes much less user and webpage history. Generally, PLC_const(10d) and PLC_const(20d) have very similar



(a) RMSD.



(b) F1 score for class 1.



(c) F1 score for class 0.

Figure 3.10 Performance comparison of different training data sizes.

performance. The former should be preferred in practice because less data are required for training.

3.3 Probabilistic Latent Class Model with Dynamic Memberships

In the work that has been done, a PLC model with constant memberships is proposed. This following work proposes an enhanced PLC model that can predict the in-view probability of an ad impression in a page view.

3.3.1 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 3.1, the memberships $P(s|u)$ and $P(p|a)$ are constant numbers learnt from training data. Before being re-trained, 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 3.11. 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.

$$P(x_{ua}|u,a) = \sum^{N_s}\sum^{N_p} P(s|f^u;\alpha_s)\cdot P(p|f^a;\beta_p)P(x_{ua}|f^{uac},s,p;w_{sp}) \tag{3.10}$$

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 3.1, 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 3.10 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 3.1 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 [12]. 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:

$$P(s|f^u;\alpha_s) = \frac{1}{Z_u}\exp(\alpha_s^T f^u) = \frac{\exp(\alpha_s^T f^u)}{\sum^{N_s}\exp(\alpha_s^T f^u)} \tag{3.11}$$

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 3.10:

$$P(x_{ua}|f^{uac},s,p;w_{sp}) = \frac{1}{\sqrt{2\pi\sigma_{sp}^2}}\exp\left(\frac{(x_{ua}-w_{sp}^T f^{uac})^2}{-2\sigma_{sp}^2}\right) \tag{3.12}$$

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 3.13 and 3.3 are the same as those between Equations 3.10 and 3.1.

$$l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\sigma}) = \sum_{u,a} \ln \Bigg(\sum^{N_s} \sum^{N_p} P(s|f^u; \alpha_s) \\ P(p|f^a; \beta_p) P(x_{ua}|f^{uac}, s, p; w_{sp}) \Bigg) \tag{3.13}$$

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

$$P(s, p|f^{uac}; w_{sp}) = \\ \frac{P(s|f^u; \alpha_s) P(p|f^a; \beta_p) P(x_{ua}|f^{uac}, s, p; w_{sp})}{\sum^{N_s N_p} P(s|f^u; \alpha_s) P(p|f^a; \beta_p) P(x_{ua}|f^{uac}, s, p; w_{sp})} \tag{3.14}$$

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

$$\alpha^*_{s\cdot} \propto \underset{\alpha_{s\cdot}}{argmax} \sum_{u,a} \left[\sum_p P(s, p|f^{ua}) \right] \cdot \\ \ln \left[\frac{1}{Z_u} \cdot \exp\left(\alpha_s^T f^u\right) \right] - \frac{\lambda}{2} \alpha_s^2 \tag{3.15}$$

$$\beta^*_{p\cdot} \propto \underset{\beta_{p_j\cdot}}{argmax} \sum_{u,a} \left[\sum_s P(s, p|f^{ua}) \right] \cdot \\ \ln \left[\frac{1}{Z_a} \cdot \exp\left(\beta_p^T f^a\right) \right] - \frac{\lambda}{2} \beta_p^2 \tag{3.16}$$

$$w^*_{sp\cdot} \propto \underset{w_{sp\cdot}}{argmax} \sum_{u,a} P(s,p|f^{ua}) \cdot \ln\left[\frac{1}{\sqrt{2\pi\sigma^2_{sp}}} \cdot \exp\left(\frac{\left(x_{ua} - w^T_{sp} f^{uac}\right)^2}{-2\sigma^2_{sp}}\right)\right] \tag{3.17}$$

$$\sigma^*_{sp} \propto \underset{\sigma_{sp}}{argmax} \sum_{u,a} P(s,p|f^{ua}) \cdot \ln\left[\frac{1}{\sqrt{2\pi\sigma^2_{sp}}} \cdot \exp\left(\frac{\left(x_{ua} - w^T_{sp} f^{uac}\right)^2}{-2\sigma^2_{sp}}\right)\right] \tag{3.18}$$

Note that the first terms of Equation 3.15 and 3.16 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.

The probability that a given scroll depth will be in view, i.e., $P(x_{ua} \geq X|u,a)$, can be calculate based on the method stated in Section 3.2.4

3.3.2 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?

Experimental Dataset 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 3.5. 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 3.5 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

Comparison Systems The performance of the proposed models is compared with several other systems: a deterministic method (DET), a logistic regression (LR) system, and a singular value decomposition (SVD) system. The details have been provided in Section 3.2.5

We also add one additional comparison system:

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 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 [10], which is a publicly available Python library.

Metrics The main metrics are still the Root-Mean-Square Deviation (RMSD), Precision, Recall, and the F1-score of class 0 (i.e., the given scroll depth is not in-view) and class 1 (i.e., the given scroll depth is in-view). The details have been provided in Section 3.2.5

In addition, we add one additional classification metric:

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.

Effect of Parameter Combination This experiment investigates 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. Grid search and random search are adopted in order to find the optimal parameter combination. For the grid search, all combinations of $N_s \in [2, 15]$ and $N_p \in [2, 15]$ are explored. For the random search, 20 combinations of $N_s \in [2, 30]$ and $N_s \in [2, 30]$ which are not included in the grid search are adopted. The range of obtained RMSDs is [0.4598, 0.4663] for the PLC_const, while that of the PLC_dyn is [0.4445, 0.4568]. 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. These combinations are used in the following experiments.

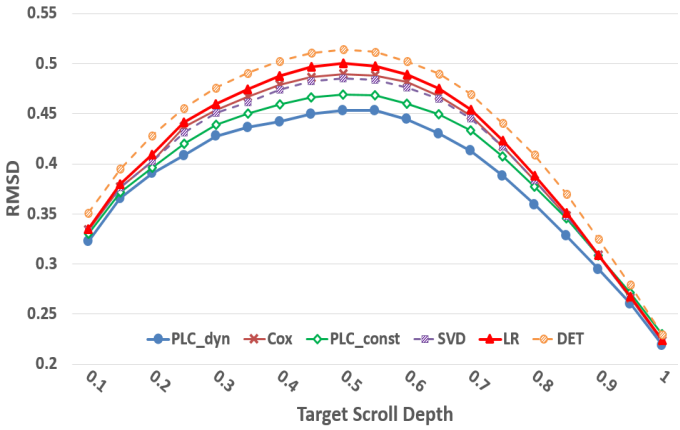


Figure 3.11 RMSD Performance

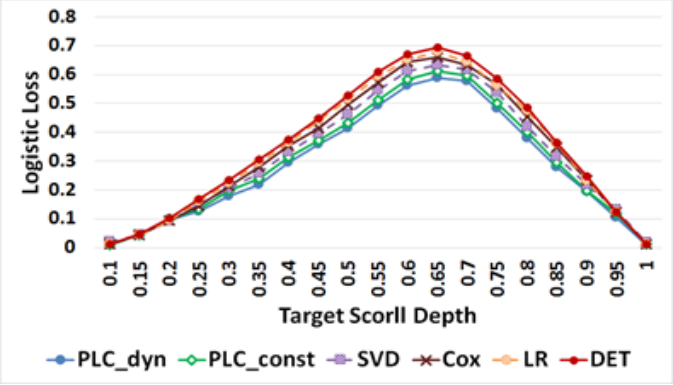


Figure 3.12 Log-loss Performance

RMSD and Log-loss Comparison The performance is measured at various target scroll depths by RMSD and Log-loss. 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 3.11 and Figure 3.12 present the results. 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]

[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.

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 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.

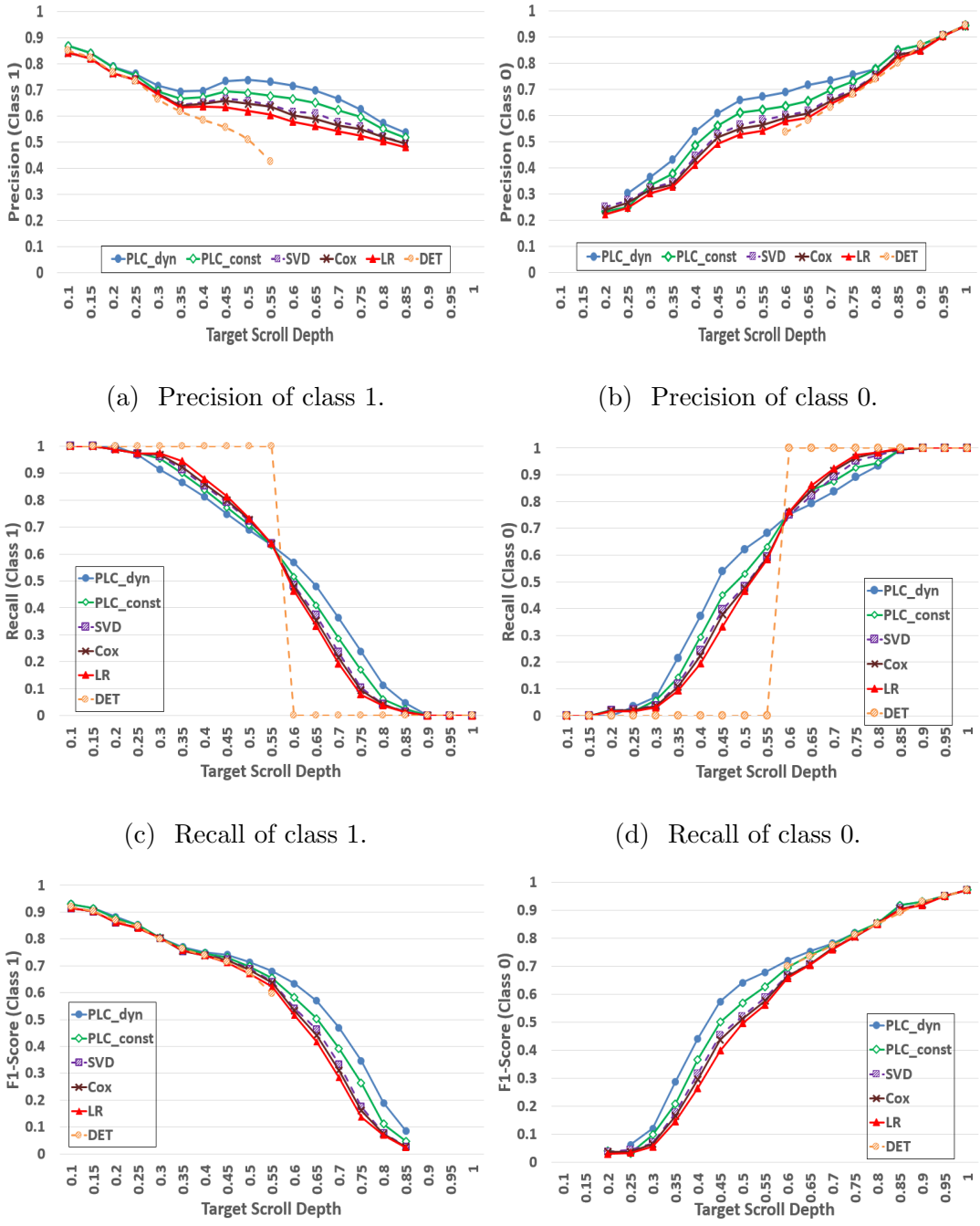
It is difficult to present the effectiveness of all features in our proposed models because there are too many sets of feature weights: Each feature in the PLC_const has $N_s * N_p$ weight vectors, while each user or page feature in the PLC_dyn has $N_s + N_s * N_p$ or $N_p + N_s * N_p$ weights, respectively. 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.

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 3.13 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



(a) Precision of class 1.

(b) Precision of class 0.

(c) Recall of class 1.

(d) Recall of class 0.

(e) F1 Score of class 1.

(f) F1 Score of class 0.

Figure 3.13 Classification performance comparison.

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 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 3.13 are not as substantial as those in Figure 3.11 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 3.13(a), 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 3.13(b). 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 [32]. 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 3.11. 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.

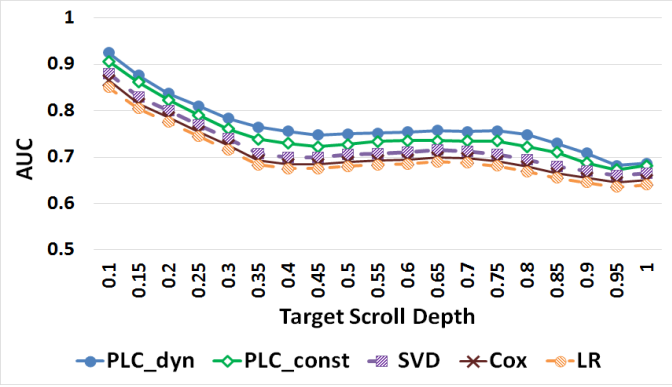


Figure 3.14 AUC comparison.

We also use AUC to evaluate the methods. Figure 3.14 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.

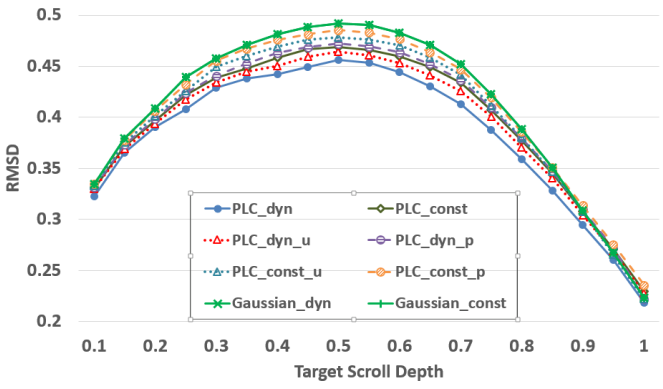


Figure 3.15 RMSD comparison by considering only latent user classes or latent page classes.

Effect of Latent 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 3.15 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 3.1 and Equation 3.10, 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.

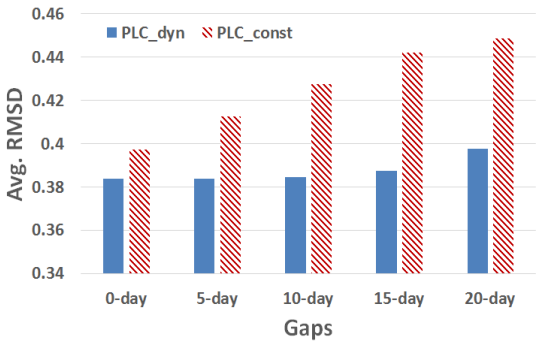


Figure 3.16 The mean RMSDs with different gaps across target scroll depths.

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

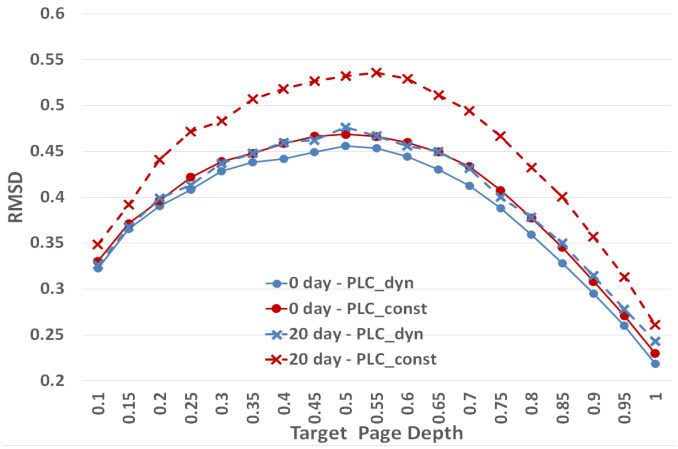


Figure 3.17 The RMSDs with 0-day and 20-day gaps.

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 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.

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 3.16 shows the average RMSDs at all target scroll depths with different gap lengths. Figure 3.17 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 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.

Performance on Different Training Data Sizes 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

Table 3.6 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)	

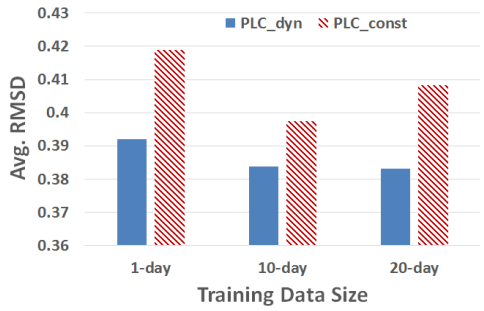


Figure 3.18 The average RMSDs with different training sizes across all target scroll depths.

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 3.6. Figure 3.18 shows the comparison of PLC_dyn and PLC_const in terms of different training data sizes. Figure 3.19 shows the comparison with 1-day and 20-day at 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

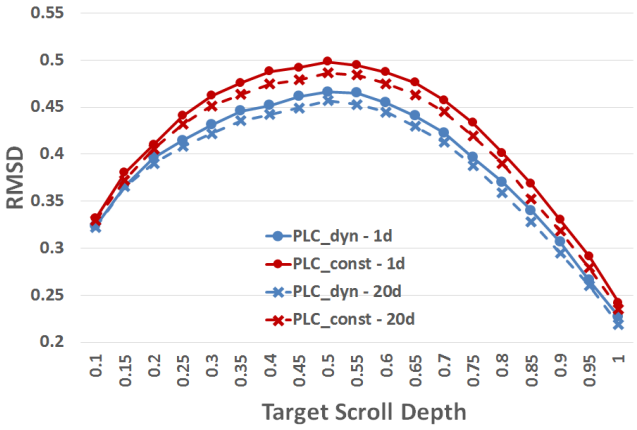


Figure 3.19 RMSDs with 1-day and 20-day training sizes.

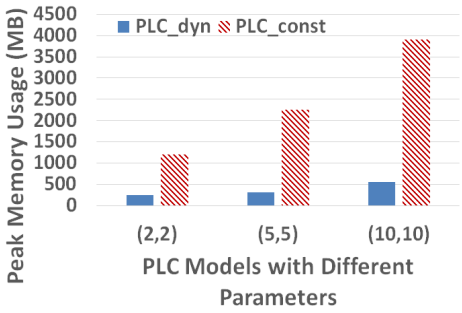


Figure 3.20 Memory comparison of training.

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.

Memory Usage Comparison Figures 3.20 and 3.21 show 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}

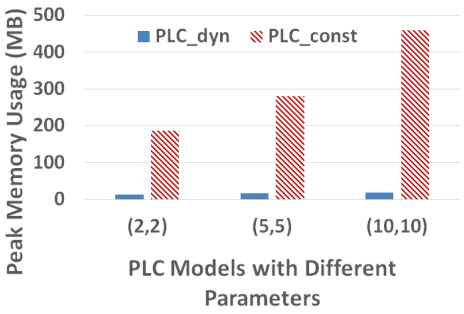


Figure 3.21 Memory comparison of testing.

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 3.3.1, $|f^u|$ is 7 and $|f^a|$ is 5.

3.4 Chapter 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.

CHAPTER 4

DWELL TIME PREDICTION

The first phase estimates ad viewability by predicting user scrolling behavior. However, it does not consider ad dwell time into account. Thus, the second phase proposes to estimate ad viewability by predicting how likely a user will stay at the page depth where an ad locates for at least certain seconds.

4.1 Problem Definition

The problem is defined as below.

Problem Definition 2. *Given a page view, i.e., a user u and a webpage a, the goal is to predict the probability that u will stay at a target page depth X for at least T seconds, i.e., X is shown on the screen for at least T seconds.*

The prediction is made after the page was requested and before the user engages with the page. The proposed methods can also be used to predict the dwell time at the target page depth.

4.2 Factorization Machines (FM) Model

4.2.1 The Real-Life Dataset

A large web publisher (i.e., Forbes Media) provides user browsing logs collected from real website visits in one week of Dec 2015 and webpage metadata. The dataset contains 2 million page views. For each page view, it records the user id, page url, state-level user geo location, user agent, and browsing events, e.g., the user opened/left/read the page. Each event stores the event time stamp and the page depths where the top and bottom of the user screen are. Once a user scrolls to a page depth and stays for one second, an event is recorded. The page depth is represented

as the percentage of the page. The reason that we adopted page percentage rather than pixels is because it provides a relative measure independent of device screen size. If a user reads 50% of a page on a mobile device, while another user reads 50% of the same page on a desktop, it can be assumed that they read the same content.

Table 4.1 is a simplified example of the user log. Each event has a time stamp so that the time that a user spent on a part of page can be calculated. To infer the current part of a page that a user is looking at, the user log also records the page depths at which the first and the last rows of pixels of the screen are. Thus, we are able to infer when a user scrolled to which part of a page and how long the user stayed. In other words, the dwell time of a page depth in a page view can be easily calculated and accumulated from the information provided by the user log.

In Table 4.1, the user scrolled to 30%-60% of the page after reading 20%-50% of the page for one minute. Thus, the dwell time of the page depths that have been scrolled past can be determined. For example, the dwell time of 20% - 30% is one minute at this moment.

Table 4.1 A Simplified Example of The User Log

User	**URL**	**Time**	**...**	**Event**	**User Behavior**
001	/abc	2/1/2015 10:00:00	...	Read Page	{"first row":20, "last row:50, ..."}
001	/abc	2/1/2015 10:01:00	...	Read Page	{"first row":30, "last row:60, ..."}

4.2.2 The Proposed FM Model

It is intuitive that the dwell time of a page depth is highly related to the user's interests and reading habits, the topic of the article in the page, the design at that page depth, etc. For instance, some users tend to stay longer on pages, while some

are less patient. A viral content may attract most users to scroll deep on the page and spend a long time on the whole page. Page depths with important topic sentences may keep most users longer on them. Thus, the characteristics of individual users, webpages, and page depths should be taken into account for depth-level dwell time prediction. More importantly, the interactions of these three factors must be modeled so that their joint effect is captured: 1) The interaction of users and pages captures a user's interest in a page. 2) The interaction of users and page depths can reflect individual users' browsing habits. For example, some users read entire pages carefully, but some only read the upper half. 3) The interaction of pages and depths models the design of individual pages at individual page depths. For example, pages that have a picture at a depth may receive relatively short dwell time at that depth because people usually can understand a picture more quickly than text. However, it is non-trivial to explicitly model user interests, page characteristics, the attractiveness of page depths, and their interactions. Also, although implicit feedback, e.g., reading dwell time, is more abundant than explicit feedback, e.g., ratings, it often has higher variability [84], which makes prediction more challenging.

Therefore, Factorization Machines (FM) [60] is adopted. Factorization Machines are a generic approach that combines the high-prediction accuracy of factorization models with the flexibility of feature engineering. The FM model has been used in applications such as context-aware rating prediction [60], retweeting [34], and microblog ranking [57]. The reason that we adopt the FM model is that it can capture the interaction of multiple inter-related factors, overcome the data sparsity, and provide the flexibility to add auxiliary information.

According to the problem definition, the basic FM model requires three factors: user, page, and page depth. The input is derived from the user-page-depth matrix built from the user logs: In the basic form of depth-level dwell time prediction, we have a three-dimensional cube containing n_u users, n_a pages, and n_d page depths.

Thus, each dwell time is associated with a unique triplet ¡user, page, depth¿. Such a 3D matrix can be converted into a list of $(n_u + n_a + n_d)$ rows. The target variable for each row corresponds to an observed dwell time represented by the triplet. N training page views lead to $N \cdot 100$ rows, as each page view contains 100 observed dwell time values (one for each percent from 1% to 100% page depth). This input is similar to what is prepared for regressions. However, regressions would not work well because the data is very sparse and they are unable to capture the interaction between the input variables.

The basic idea of FM is to model each target variable as a linear combination of interactions between input variables. Formally, it is defined as following.

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \langle \mathbf{v_i}, \mathbf{v_j} \rangle \, x_i x_j \tag{4.1}$$

where, $\hat{y}(\mathbf{x})$ is the prediction outcome given an input $\mathbf{x}$. w_0 is a global bias, i.e., the viewability of the page depth or the overall average depth-level dwell time. $\sum_{i=1}^{n} w_i x_i$ is the bias of individual input variables. For example, some users would like to read more carefully than others; some pages can attract users to spend more time on them; some page depths, e.g., very bottom of a page, usually receive little dwell time. The first two terms are the same as in linear regression. The third term captures the sparse interaction between each pair of input variables.

Unlike standard regression models which model the weight of each interaction by a real number w_{ij}, the FM model uses a factorized parametrization to capture the interaction effect (Eq. 4.2). Such low-rank interaction allows the FM model to estimate reliable parameters even in sparse data.

$$\langle \mathbf{v_i}, \mathbf{v_j} \rangle = \sum_{k=1}^{K} v_{ik} v_{jk} \tag{4.2}$$

The basic FM model works with only three factors: user, page, and depth. However, context information can also help improve the prediction performance. Thus, we identify three context features, *viewport* (i.e., the part of a user browser visible on the screen), *local hour*, and *local day of the week* (denoted by weekday in the experiments), which are likely related to user reading behavior. The viewport indicates the device utilized by the user (e.g., a mobile device usually have a much smaller visible browser area than a desktop) and can directly determine the user experience. Specifically, one viewport value consists of the height and the width of a browser, e.g., 1855×1107. To reduce sparsity, both heights and widths are put into buckets with size 100 pixels. For instance, 1855×1107 can be discretized into 18×11. The local hour and local day of the week, expected to reflect if users are working, are inferred from the GMT time stamp and user geo provided in the user log.

In addition, although in theory user demographics and page attributes are already considered in the latent user and page dimensions, incorporating these additional sources of information as features may further improve the prediction accuracy in some applications [37]. For user demographics, we consider user geo locations because this is the only explicit feature about users that can be easily obtained by publishers without violation of user privacy. User geo, inferred from IPs, may reflect a user's interests and education, and it may determine the user's network condition. Specifically, geo is the country name if the user is outside USA or a state name if she is within USA.

For page attributes, we consider article length, channel, and freshness. Article length is represented by the word count of the article in the page, and it has been proven to be a significant factor impacting page-level dwell time [83]. However, its influence on page-depth-level dwell time is still unclear. Article lengths are put into buckets so that there are a limited number of possible states. The channel of the article in a page is its topical category on the publisher's website, e.g., finance and

lifestyle. A channel can be considered as a high-level topic label of a page. Freshness is the time span between the page is read and the page is firstly published on the website. Freshness is measured by days. The freshness of an article may determine the interests of a user on it. Fresh news may receive more user engagement.

The viewport content is also models by several state-of-the-art models because it is believed that the content shown in a user's browser affects the time that the user spends on it. The user log records the position of each viewport and the article meta data include the content of each article. Thus, it is possible to obtain the textual content shown in the user's browser. Several models are used to model the semantics of each viewport content: TF-IDF, LDA, and Doc2Vec.

TF-IDF [79], short for term frequency-inverse document frequency, is a very commonly-used method to weight words based on their importance to a textual document in a collection. The TF-IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. In our application, we first training a TF-IDF model over all training articles. In the test process, given a viewport content, the keywords with high TF-IDF values are extracted. These keywords form a vector to represent the topic of the viewport content.

LDA [74], short for Latent Dirichlet Allocation, is an unsupervised process for inferring the topics in a textual document. It outputs a clearly-defined probability for arbitrary documents. As a generative mode, LDA provides the mechanism for finding patterns of term co-occurrence and using those patterns to identify coherent topics. in particular, all terms that co-occur with term w are more likely to have been generated by the topic that w belongs to. Each word in a document can generate different topics with probabilities. Containing many different words, each document can be considered as a mixture of a few of topics and that each word's creation is

attributable to one of the document's topics. The parameters in a LDA model can learned from the training corpus. More details about how LDA works can be found in [8]. Since the webpage articles in our corpus are relatively long (compared to short text, e.g., tweets), it is suitable to use LDA to model the topic distribution of a document due to the presence of abundant word co-occurrence. Thus, all training articles are fed into the LDA model. The learned model can be use to infer the topic distribution of each test viewport content. In the experiments, we compare two different ways to incorporate LDA outcome into the FM model. The first is to only consider the latent topic with the highest probability and concatenate it with other feature using one-hot encoding. In other words, one viewport content is assigned to only one topic. This strategy is often used in topic modelling. The second strategy is to consider all latent topics. The topic distribution vector will be concatenated with other features. In addition, we evaluate different pre-specified numbers of latent topics in the experiments.

Doc2Vec [41] is an unsupervised learning of continuous representations for variable-length pieces of texts, such as sentences, paragraphs or entire documents. Unlike traditional text representation schemes, e.g., TF-IDF, Doc2Vec take into account the ordering and semantics of the words. For instance, "car" and "auto" are treated as two totally different words, while they are regarded as synonyms in Doc2Vec. Given a chunk of text, Doc2Vec provides a fixed-length feature vector to represent the meaning of the text. The vector can be used as an input in the FM model. Doc2Vec is extended based on the Word2Vec algorithm [48]. Word2Vec is a type of shallow two-layer neural networks that are trained from training data to produce word embeddings. Instead of relying on the number of co-occurrence as what LDA does, Word2Vec takes the words in a context with a fix-size window as input and understands a word by predicting its surrounding context (cBoW) or predicting a word given it surrounding context (skip-gram). Skip-gram model is adopted in

this project since existing work [48] shows that it outperforms the other variants on analogy tasks. The output of Word2Vec is word vectors that can be mapped into a vector space such that semantically similar words have similar vector representations, e.g., the word vectors of "car" and "auto" are close.

In Doc2Vec, the vector representation of a chunk of text is trained to be useful for predicting words in a paragraph. More precisely, Doc2Vec concatenates the paragraph vector with several word vectors from a paragraph and predict the following word in the given context. Similar to word vectors, the Doc2Vec vectors of two pieces of text which have close meaning should be very close to each other. Given a unseen piece of text, a fully trained Doc2Vec model can infer a vector to represent its meaning. The Doc2Vec used in this project is developed based on Gensim [59]. All training articles are fed into the Doc2Vec model. The learned model can be use to infer the feature vector of each test viewport content. We evaluate different dimensionalities of the feature vector in the experiments.

4.2.3 Evaluation

Experiment Datasets A one-week user log is split into three sets of training and testing data. The experimental results are reported by taking the average over the sets. On average, the training and test data contain 150K+ and 20K+ page views, respectively. The training/test data consist of all depths of all training/test page views.

Comparison Models Several comparison systems are developed as following:

GlobalAverage: In dwell time prediction, i.e., Section 4.2.3, it computes the average dwell time of each page depth X in all training page views. If a user did not scroll to X before leaving the page, its dwell time in the page view is zero. In viewability prediction, i.e., Section 4.2.3, it computes the fraction of training page depths whose dwell times are no less than the required dwell time. In both tests, 100

constant numbers are obtained after iterating over all training pageviews. They are used to make a deterministic prediction for the corresponding page depth.

UserAverage: It is similar to GlobalAverage. But it computes the average dwell time of each depth X based on each user's reading history (rather than all training page views). In viewability prediction, for a depth of a training pageview, whether or not it is viewed for at least certain seconds is recorded, i.e., 0 or 1. The probabilistic prediction is made based on the average over all binary outcomes of a page depth of a user.

PageAverage: Similar to UserAverage, it computes the average dwell time of each depth X based on each page's history.

Regression: Two regression models are built. 1) The first, *Regress_bc*, is developed based on existing work on page dwell time prediction [83]. To apply to depth-level prediction, one more feature, i.e., page depth, is added. In particular, topical categories are represented by channels. Page length is calculated by article word counts. Device types are identified from user agents. In the viewability prediction test, logistic regression with the same features is adopted because it can output probability that the dwell time of X is at least certain seconds. 2) The second, *Regress(depth+dov2vec_150+channel+viewport)*, is developed based on the finding in Section 4.2.3 that shows the viewport, doc2vec_150, and channel are the best features for improving prediction. To predict at page depth-level, depth is added as well. For both models, in the viewability prediction test, logistic regression with the same features is adopted. They outputs the probability that the dwell time of X is at least certain seconds.

Metrics The metrics we adopt are Root-Mean-Square Deviation (RMSD) and Logistic Loss. Both serve to aggregate the magnitudes of the errors in predictions for

various times into a single measure of the predictive power of a method. Thus, for both metrics, lower values are better.

RMSD: The RMSD has been widely used in regression problems. It measures the differences between the values predicted by a model, $\hat{y}_i$, and the values actually observed, y_i. For depth-level dwell time prediction, it is defined as the square root of the mean square error:

$$RMSD = \sqrt{\frac{\sum_{j=1}^{N}\sum_{i=1}^{100}(\hat{y}_{ij} - y_{ij})^2}{N \cdot 100}}$$

where N is the number of test page views. The second sigma accumulates the errors at all 100 page depths in the ith page view. y_{ij} is the actual dwell time, at the jth page depth in the ith page view. $\hat{y}_{ij}$ is the corresponding predicted dwell time.

Logistic Loss: It is widely used in probabilistic classification. Compared to the RMSD, it penalizes a method more for being both confident and wrong. For example, if for a particular observation, a classification model assigns a very small probability to the correct class then the corresponding contribution to the Log Loss will be very huge. In our case, the probability is interpreted as how likely it is that the dwell time of a page depth is at least a certain amount of time.

$$logloss = -\frac{1}{N \cdot 100}\sum_{i=1}^{N}\sum_{j=1}^{100}[y_{ij}\log(\hat{y}_{ij}) + (1 - y_{ij})\log(1 - \hat{y}_{ij})]$$

Comparison of Feature Combinations We add context and auxiliary features, including user features, page features, and depth features, into the basic FM model in order to evaluate the effect of different combinations. The models are applied to predict the dwell time of every page depth in each test page view. Since it is unknown which feature combination is the best, we first add one more feature to the basic FM model and keep adding more features to the best one once at a time. The result of

Table 4.2 The RMSD Comparison by Adding One More Additional Feature

Feature Groups	**Models**	**K=10**	**K=20**	**K=30**
Basic	FM	12.5707	12.5401	12.5420
Context	FM (weekday)	12.6575	12.6665	12.6779
	FM (hour)	12.7563	12.8792	12.8262
	FM (viewport)	12.3645	12.3285	12.2909
User	FM (geo)	12.5465	12.6186	12.5971
Article	FM (length)	12.636	12.6642	12.6388
	FM (channel)	12.3752	12.3597	12.3489
	FM (freshness)	12.4770	12.4944	12.6223
Viewport Content	FM (TF-IDF)	12.6404	12.6239	12.6474
	FM (topic_10)	12.5308	12.5335	12.6072
	FM (topic_20)	12.3929	12.5092	12.5122
	FM (topic_30)	12.5168	12.5719	12.5184
	FM (topic_40)	12.7689	12.6897	12.7385
	FM (topic_group_10)	12.3852	12.3416	12.2912
	FM (topic_group_20)	12.4913	12.4687	12.4393
	FM (topic_group_30)	12.4298	12.3899	12.4137
	FM (topic_group_40)	12.4576	12.4065	12.3866
	FM (doc2vec_50)	12.4584	12.4298	12.3042
	FM (doc2vec_100)	12.4014	12.3790	12.2831
	FM (doc2vec_150)	12.3082	**12.2675**	12.2965
	FM (doc2vec_200)	12.3271	12.3498	12.3525

adding one additional feature is presented in Table 4.2. The performance is measured by RMSD. The first row is the basic FM model, which has only three dimensions, i.e., user, page, and depth. We also vary the dimension of the 2-way interactions, K, which is the length of the latent vector v for each variable (Equation 4.2).

The result shows that some features can significantly improve the prediction performance of the basic FM. Viewport is the most significant context feature. Intuitively, viewport indicates the type of device, which influences reading experience and thus the way users engage with webpages. Channel is also a significant one in that, representing the topic of the whole page, it directly determines the interest of the user to the page. The Channel information is provided by the creators of the article metadata. Thus, it can be considered as 100% correct.

Nevertheless, some features cannot help on prediction. For instance, adding weekday or hour cannot decrease the RMSD of the basic Fm model. This indicates that probably the time that one user spends on a page depth does not significantly differ by the hour of the day and the the day of the week. Also, User geo location does not enhance the performance of the basic FM. The possible reason is that the granularity of user geo is too coarse. In the user log, user geo is state-level if in USA, otherwise country-level. Learning latent features for each geo cannot specifically capture the characteristics of individual users. For example, it can be imagined that one from Buffalo and one from New York City may have different interests and reading behavior, even though both are in New York State. Article length may not play a key role at depth-level, as the text length in a screen is determined by the viewport size, not the length of the entire article.

Four methods are adopted to model viewport content. FM (TF-IDF keywords) considers all non-stopwords with high TF-IDF in a viewport content. FM (topic_n) considers the most probable topic calculated by LDA with the topic number n. FM (topic_group_n) considers the topic distribution calculated by LDA with the topic

number n. In contrast to FM (topic_n), FM (topic_group_n) takes into account all latent topics whose probability is more than 0. The value of the possible topics are its probability, not binary. In this way, the most probable topic is still weighted higher than others. Hence, it is expected that FM (topic_group_n) can provide more details about the topic of a viewport content. Lastly, FM (doc2vec_m) uses a Doc2Vec vector to model the content of a viewport. We vary the length of the vector m to see its impact on the performance. Table 4.2 shows that the RMSD of FM (TF-IDF keywords) is not as low as the basic FM. The possible reason is that the keywords of a viewport content still contain much noise. Also, keywords are extremely sparse: Most keywords only occur in one viewport content. The performance of FM (topic_n) varies according to the pre-defined topic number n. $n = 20$ leads to the best prediction performance. Compared to FM (topic_n), FM (topic_group_n) generates more stable performance because it consider not only the topic with the highest probability, but also all other topics with a probability more than 0. It can more smoothly reflect the topical relatedness among different viewport content. Finally, considering the deeper relationship among words, FM (doc2vec_m) on average has the best performance compared to the other three methods. FM (doc2vec_150) with $K = 20$ reaches the best performance.

Increasing K does not always lead to performance improvement. Longer latent feature vectors may fit the data better, while in some cases may cause overfitting. As the Bias-Variance trade-off, the optimal K can be obtained by cross-validation.

In order to further explore the best performance, more than one features are Incorporated into the basic FM model at the same time. Since FM (doc2vec_150) with $K = 20$ reaches the lowest RMSD, additional features are kept adding into it. Also, as doc2vec models viewport content, the other features, i.e., TF-IDF keywords and LDA, fall into the same category are not considered this time. For example, since

dov2vec_150 already models the content of a viewport, keywords and LDA topic will not be considered to be added into FM (dov2vec_150).

Table 4.3 The Comparison by Adding More Additional Features to FM (doc2vec_150)

Models	**RMSD;K=20**
FM (dov2vec_150+viewport)	12.2301
FM (dov2vec_150+channel)	12.0733
FM (dov2vec_150+freshness)	12.3487
FM (dov2vec_150+channel+viewport)	**12.0419**
FM (dov2vec_150+channel+freshness)	12.1985
FM (dov2vec_150+channel+viewport+freshness)	12.1827

Table 4.3 shows that the FM model with dov2vec_150, channel, and viewport as additional features gets the lowest RMSD, i.e., the best performance. In other words, the dwell time of a given depth is determined by the content around that depth (captured by dov2vec), the topic of the whole article (captured by channel), and the size of the browser (captured by viewport). Freshness is not as predictive as the other three. The possible reason is that, although fresh news attract a great deal of attention, users sometimes spend long time on stale articles because they mean to seek the information in those stale articles.

Page Depth-level Dwell Time Prediction We compare the best model obtained from the above experiment, i.e., FM (dov2vec_150+channel+viewport) with K=20, with the comparison systems. All models are applied to predict the exact dwell time of each page depth in test page views. The results in Table 4.4 demonstrate that the FM model significantly outperforms the comparison systems. This is because it is able to overcome sparsity and capture pairwise interactions between features. The

Table 4.4 Depth Dwell Time Prediction Comparison

Approaches	RMSD
GlobalAverage	13.6971
PageAverage	13.5243
Regress_bc	13.2643
UserAverage	13.1482
Regress (depth+dov2vec_150+channel+viewport)	12.9043
FM (dov2vec_150+channel+viewport;K=20)	12.0419

RMSDs of PageAverage and UserAverage are better than GlobalAverage because their predictions are tailored to each page or each user. Also, the result indicates that controlling the user variable seems to be more effective than controlling the page variables. This is because that, although both variables matter, dwell time is more determined by individual users' subjective wills. The RMSD of Regress_bc is not as low as the one of UserAverage, which indicates that methods for page-level dwell time prediction cannot be easily applied to depth-level prediction. Without capturing the interaction of features, Regress(depth+dov2vec_150+channel+viewport) does not obtain as good prediction outcome as FM(dov2vec_150+channel+viewport;K=20).

The result shown in Table 4.4 are calculated over all test page depth. In order to look into the performance at different areas of pages and evaluate the robustness of the proposed method, page depths are split into different buckets: bucket1: [1%, 25%], bucket2: [26%, 50%], bucket3: [51%, 75%], and bucket4: [76%, 100%]. According to the result shown in Table 4.1, the proposed method stably outperforms the others in all buckets.

Generally, the prediction error is decreasing with the increase of the page depth. The reason is that most users only read the first half of the page. Therefore, the dwell time of the page depths near the bottom of the page is mostly zero second. It is easier

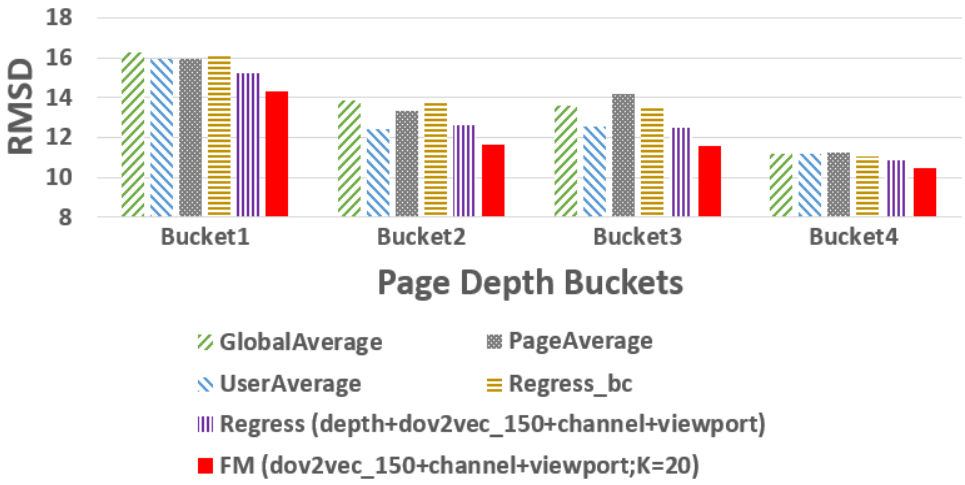


Figure 4.1 Depth dwell time prediction comparison (buckets).

to do prediction at the bottom of the page. This is why the prediction performance of all methods are closer in the bucket4, while the proposed method is still the best.

Viewability Prediction Viewability can be regarded as the probability that an item (e.g., an ad) at a page depth will be viewable. This can be treated as probabilistic classification. Therefore, we run an experiment to evaluate whether the FM model can handle this problem.

We vary the dwell time threshold of a viewable impression from 1s (IAB standard) to 10s. The target variable of each page depth in the dataset is 1 if its dwell time is at least T seconds; otherwise 0. In this way, the prediction problem is converted from regression to classification. The prediction outcome of each test page depth is the probability that its dwell time is at least T seconds.

Figure 4.2 shows that the FM model clearly outperforms the baselines. It is also noticed that the FM model achieves the best performance at the two ends (1s and 10s). Given a page depth, it is more challenging to predict if the dwell time is at least 5s. The reason is that the number of page depths with dwell time at least 5s and the number of page depths with dwell time less than 5s are very close (about 50%). In contrast, there are about 70% page depths whose dwell time is at least 1s. Similar

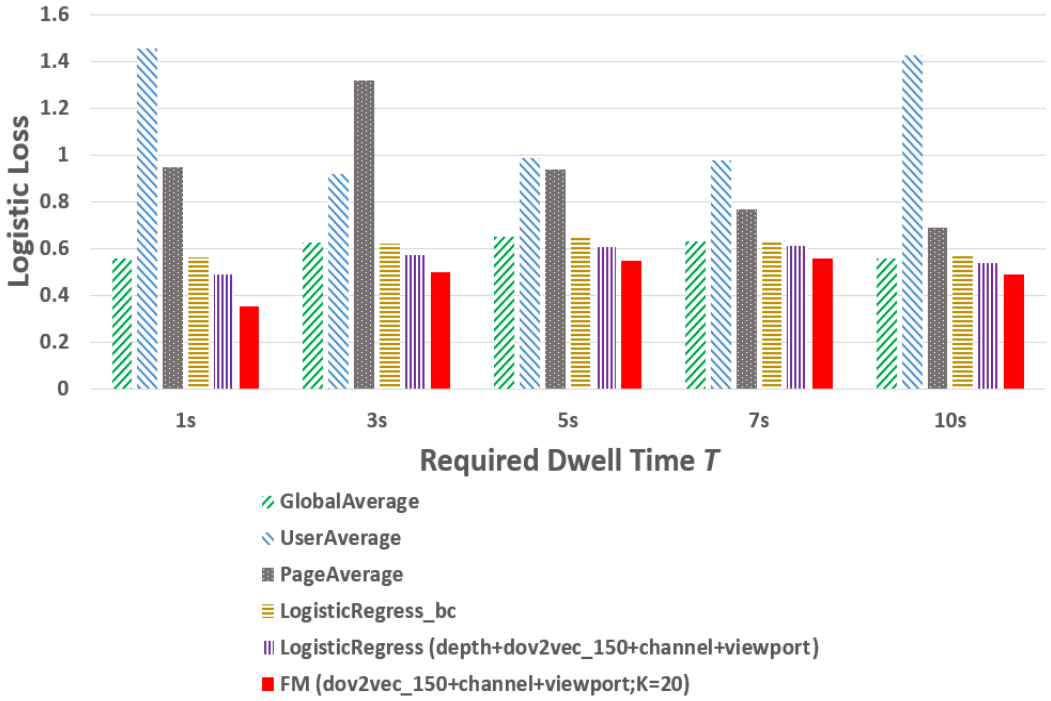


Figure 4.2 Viewability prediction comparison.

to the depth-level dwell time prediction, GlobelAverage and LogisticRegress_bc have similar performance. Also, the LogisticRegress with significant features is better than other baselines.

One interesting observation is that, although UserAverage and PageAverage outperform GlobalAverage by RMSD, as shown in Table 4.4, they are much worse than GlobalAverage by logistic loss in viewability prediction. Also, they do not have as stable performance as other methods. The main reason is that most users and pages in the test data have few historical pageviews in the training data. Also, most pageviews have sparse dwell time distribution, i.e., the dwell times of many page depths are 0. In this case, for individual users or pages, the viewability predictions of a depth are close to 0 or 1. Once the prediction is incorrect in the test data, the penalty by logistic loss will be large because it heavily penalises classifiers that are confident about an incorrect classification. For instance, the dwell times at 10% depth of a user's all historical pageviews are 0s, 0s, 0s, and 3s. In a test pageview, the user

spent 1s at that page depth, which is the ground truth. Given $T = 1$, the prediction of the user at this depth will be always $(0 + 0 + 0 + 1)/4 = 0.25$. This means the classifier thinks that the depth will very likely be viewed for less than 1s. However, the logistic loss of the prediction is $logloss(0.25, 1) = 25.9047$, which is a very huge penalty. This is because that in classification problem it is better to be somewhat wrong than emphatically wrong. This characteristic is very important for publishers because it can help publisher avoid large decision-making errors, for instance, they are suggested that an impression at a page depth will be certainly viewable, but it turn out to be an unviewable one.

4.2.4 Feature Analysis

We also look into some features and investigate how user reading behaviors are related with the feature values. This may influence advertisers' biding behaviors as well as publishers' ad allocation strategies and website design.

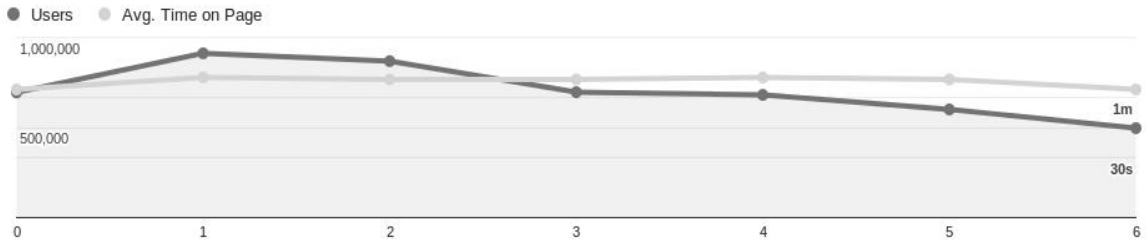


Figure 4.3 Day of week vs. traffic and mean page-level dwell time (New York State; 0=Sunday).

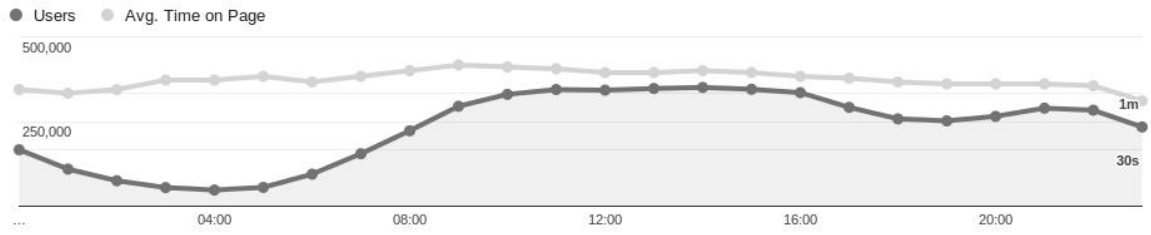


Figure 4.4 Hour of day vs. traffic and mean page-level dwell time (New York State).

Weekday and Hours We investigate whether user reading behavior varies with time, i.e., (local) day of week and (local) hours of day. The long-term data are provided by Google Analytics (GA)[1]. Forbes uses it to track and report traffic and the usage of websites. Since the time recorded in GA is the visit's time converted to the timezone configured for the GA profile (Forbes profile uses the US Eastern Time), we fix the region of the visits to the New York State. The data provided by GA is collected in the most recent one month, i.e., Aug 2016.

Figure 4.3 shows website traffic and the mean page-level dwell time on different days of week. It only includes the pageviews from the New York State in order to eliminate the impact of different timezones. Although website traffic varies by the days of week, the mean page-level dwell time almost does not have any fluctuation. Users spend the same time on the pages on different days of week. Besides, Figure 4.4 presents a more obvious pattern that the page-level dwell time does not vary by the change of the hours. Concretely, more users come to Forbes in the daytime. But they do not seem to spend more time on pages. We also tried multiple regions at different timezones, e.g., California. The findings are the same.

In addition, in order to reveal how much time is spent at different page depths, we go back to our user log data of Dec 2015 and sample at least 100,000 pageviews from each weekday/hour. Then, the dwell time distributions over all page depths on representative weekdays/hours are plotted in Figures 4.5 and 4.6. These weekdays/hours are either the time point that have either the longest or the shortest mean page-level dwell time. Similar to the page-level dwell time, the depth-level dwell time does not influenced much by time as well. The difference of the peaks are only about three seconds. The shape of the dwell time distributions are also very close to each other on different weekdays/hours. The possible reason is that, although most users tend to visit the website in the daytime of weekday, the users who visit during

[1]https://analytics.google.com/

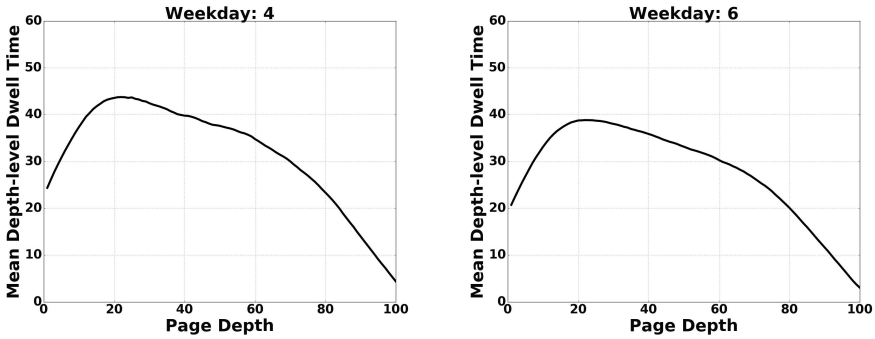


Figure 4.5 The comparison of mean depth-level dwell time on Wednesday and Saturday (in seconds).

the midnight or weekend are meant to seek for certain information on the pages. In this case, they may not spend much shorter time than the ones visit in daytime of the weekdays.

Studies [87, 77] discover that, in the current pay-by-impression pricing model, the wining bidding prices significantly vary by the hours of day. Specifically, the wining bids peak at 8-10am due to intensive competition. However, our research finds that the page depth-level dwell time does not vary much. Users may still engage a lot with web articles at midnight. Thereby, the chance that ads exposed on screen for long enough time at midnight is as the same as that in the daytime. Namely, ad viewability may not change largely across different hours of day and days of week. Therefore, through this research, advertisers hopefully can realize that the impressions at midnight do not have much lower viewability. Hence, they do not necessarily compete with each others in the daytime and consequently pay higher prices for the marketing chances that they can also get during non-peak time.

Channels In each of the six primary channels on Forbes website, 2000 pageviews are randomly sampled. For each pageview, the dwell time of the user spent on every page depth is calculated. Thus, each pageview has a vector of length 100. Each value in the vector is the time that the user spent on the corresponding page depth.

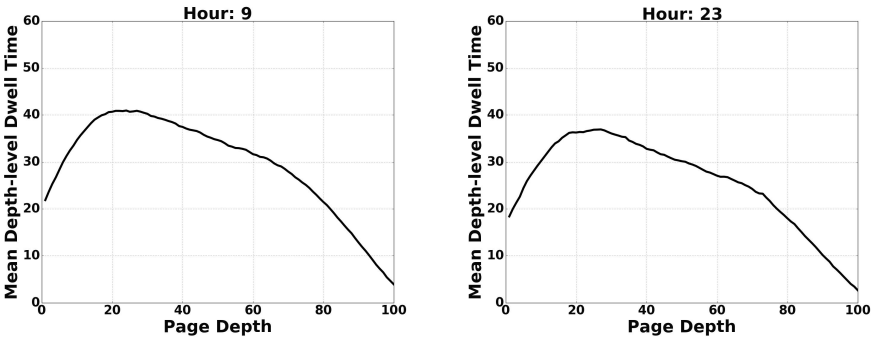


Figure 4.6 The comparison of mean depth-level dwell time on different hours of day (in seconds).

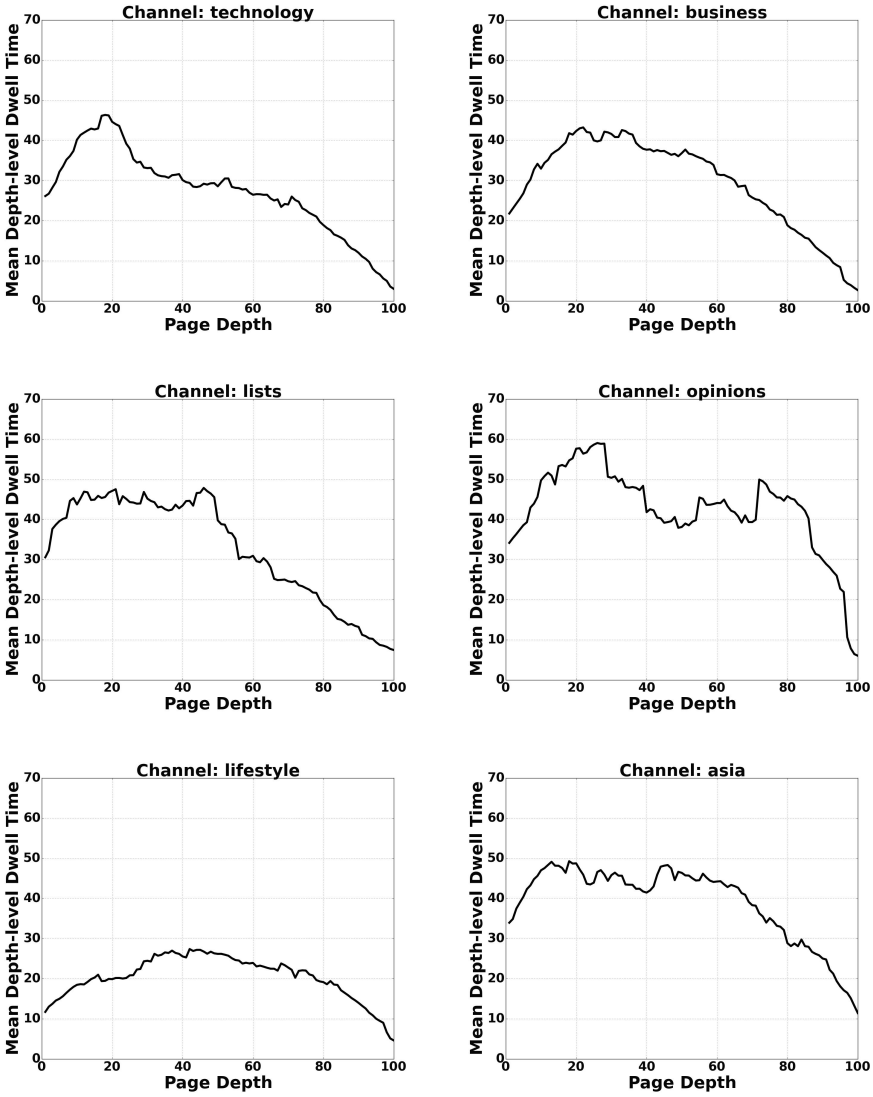


Figure 4.7 The comparison of mean depth-level dwell time across channels (in seconds).

For each channel, the centroids of the 2000 vectors are calculated by averaging. The resultant 6 centroids can be considered as summaries of the dwell time patterns of corresponding channels. The centroids are plotted in Figure 4.7.

All six plots indicate that users usually spend more time on the first half of the page than the second half. Also, the top several percents of the page are usually skipped because this area is always the menu bar. However, the patterns of individual channels are not identical. Users tend to spend less time on the lifestyle channel, which usually publishes web articles about travel, sports, and autos. Intuitively, users may not read every single sentence in these pages. On the other hand, users spend long time on the opinion channel, which publishes updated analysis on popular news. It is reasonable in that these opinion articles are original and can attract users to read about the authors' points. Likewise, as the most well-known product, the lists channel, which usually publishes the rankings, e.g., the The World's Billionaires, receive high engagement on the first half, while users quickly lose attention at the second half. The possible reason is that most users only focus on the top positions when reading a list, e.g., people more care about how are the top three richest billionaires than the top tenth. Surprisingly, it is revealed that users stay longer only in the very top part of the technology articles: people would like to know what is happening in the technology industry, but have not interest in the detail about the technology. In addition, while business and Asia share very similar patterns across page depths, Asia receives slightly longer dwell time. Publishing articles about the economy and billionaires of Asia, the Asia channel has significantly more Asian visitors. Due to the language barrier and relatively slow network connection, Asian visitors usually spend relatively more time on pages.

While the figures show on average the users tend to have different levels of engagement in different channels, the difference will be more outstanding by looking into individuals because users have diverse interests. By recommending articles of

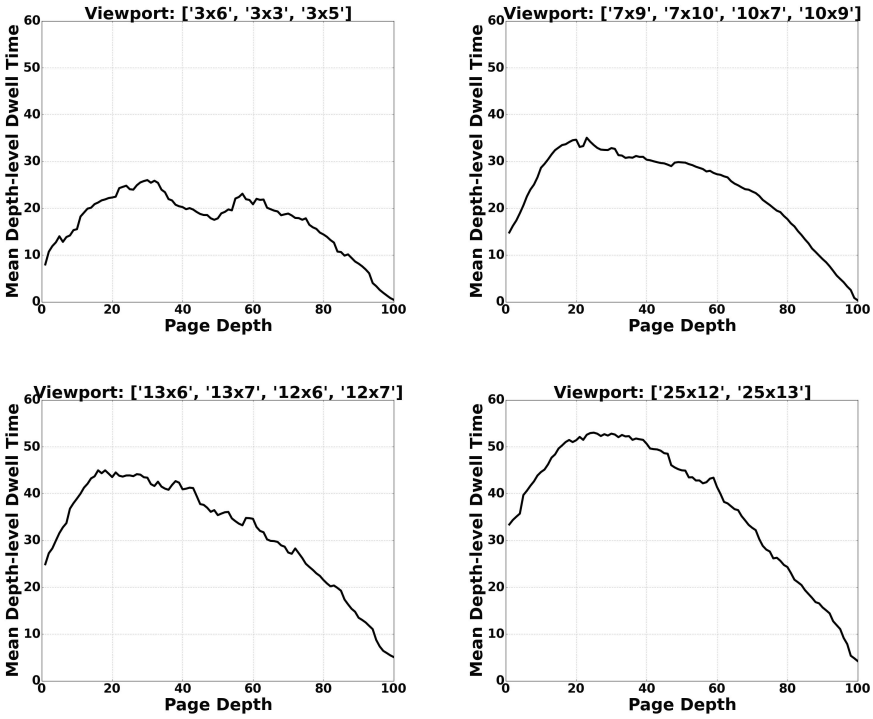


Figure 4.8 The comparison of mean depth-level dwell time across viewport categories (in seconds).

interest to specific users, publishers can boost the engagement of individual pageviews, attract higher bidding prices, and thereby obtain more ad revenue. In future work, we will uncover the relationship between user engagement and ad revenue.

Viewport We also investigate user reading behaviors on different viewports. As mentioned before, viewport is the user's visible area of a web page. Viewport is highly determined by the size of the user's browser. It can also indicate the devices that the user is using, e.g., a very small viewport size indicates that the user probably is using a mobile device. Since it is possible that users may adjust their browser into many different sizes, we group viewport sizes by every 100 pixels. For example, '320x520' is represented as '3x5'.

Only popular viewport sizes are considered in this experiment. According to an online public resource[2], viewport sizes are grouped into four categories which represent four main display devices: 1) '3x6', '3x3', and '3x5' frequently occur in the dataset among all small viewports. In fact, these are probably mobile visitors. 2) '7x9', '7x10', '10x7', and '10x9' are significant in the middle size viewports. These are probably tablet visitors. The first two correspond to the portrait mode, while the last two corresponds to the landscape mode. Unlike on mobile devices, the number of landscape viewports are almost as significant as the portrait mode. 3) '13x6', '13x7', '12x6', and '12x7' may correspond to the viewports on laptops. 4) We also collect data for big screens: '25x12', '25x13'. It is observed from the user log that a number of users visit the website using a big monitor. In each category, 2000 pageviews are randomly sampled. The result is shown in Figure 4.8.

People generally spend less time on mobile devices. Existing research shows that people usually use mobile devices for casual reading [24], where people use the mobile Web to access general information without a specific goal. In this case, users may not stay long on pages. Also, the dwell time distribution of mobile devices seemingly has two peaks: one is near 30%, the other is near about 60%. The reason may be that flicking fingers on the screen is as easy as scrolling the wheel of a mouse [39]. People usually read the first paragraph and then quickly flick to the last part of the article. In contrast, the dwell time distribution of tablet devices is smoother because a tablet have a bigger screen than the a mobile device. Thus, when a user is reading the first/last paragraph, the middle part is also in-view. In this case, the dwell time in the middle of a page is not significantly lower than that in the two tails.

Existing work [64] finds that users take less time to finish assigned search tasks on devices with bigger displays. However, it is not also held in out case that depth-level dwell time on bigger screens is lower than that on smaller screens: According to

[2]http://viewportsizes.com/. The owner of this website collects the viewport sizes on many devices. For example, the typical viewport size of iPhone 6 is '375x667'

Figure 4.8, dwell time is increasing with the increase of the viewport size. The main reason is that bigger viewports can display more content. Therefore, users would stay longer to read on the depths displayed in the viewports without much scrolling.

Low engagement may lead to low ad viewability. With the ubiquity of mobile devices, a rapidly increasing number of visits are from mobile devices. Low engagement on mobile devices may hurt the quality and value of publishers' impression inventory. Therefore, publishers should improve the usability of their website, especially on mobile devices. One possible way is to present a mobile-friendly webpage layout on mobile devices. Improving user experience, publishers may keep users longer on the webpages.

4.3 Deep Sequential Neural Networks

4.3.1 Introduction

Although the FM model uses latent vectors to model input variables, it is still insufficient to capture the deep pattern among input variables.

Therefore, this work considers webpage depth viewability prediction as a sequential labeling problem. Three deep learning networks are proposed. They can predict how likely it is that a given page depth will be viewed by a user for at least a certain dwell time. The proposed models utilize the information of the previous page depths to predict the viewability at the current page depth. In particular, the models leverage Recurrent Neural Network (RNN) using the Long Short-Term Memory (LSTM) to model sequential dependency into predicting webpage depth viewability.

Since users read webpages from top to bottom, the dwell times of all page depths in a page view forms a sequence of inputs. Therefore, RNNs can be adopted in our case because they leverage the internal memory to process sequences of inputs. However, traditional RNNs suffer from the vanishing gradient problem [29]. LSTM

RNNs are designed to avoid the long-term dependency issue by adding gates to control how much past information is transferred through time steps. The problem is modelled as a 'many-to-many' sequence labeling problem. In the first proposed model, LSTM_noInteract, every time step outputs one prediction outcome, i.e., the viewability of a page depth. The input of each time step in the proposed LSTM RNN contains information about the user, the page, the depth, and the context. The first three are learned using three embedding layers. Since user behavior is determined by the interaction of user, page and depth. In the second proposed model, LSTM_Interact, we also propose to consider the interaction of user, page, and depth by multiplying their embedding vectors before sending the information to the LSTM layers. Furthermore, users often scroll-back on pages. The time a user spent at lower page depths also may indicate the time the user will spend at upper page depths. In addition, in single directional LSTM, predictions made at very top page depths rely on few previous page depths. For instance, only the LSTM layers at the page depth of 1% can contribute to the prediction at the page depth of 2%. In this case, the end prediction performance will be discounted. In the third model, Bi-LSTM_Interact, we therefore upgrade LSTMs to bi-directional LSTMs, which can take future information, i.e., lower page depths, into account. In this case, all page depths will help the prediction at 2%.

The models are evaluated using real data from Forbes Media, a large web publisher. The experimental results demonstrate that our models outperform the comparison models, i.e., GlobalAverage, Logistic Regression, and Factorization Machines. The model with the best performance is Bi-LSTM_Interact that is powered by bi-directional LSTMs and considers embedding interaction.

To summarize, the main contributions are:

- To the best of our knowledge, we are the first to use many-to-many LSTM networks, i.e., each timestep generates one output prediction, to predict user

behavior (existing work uses time-series LSTM, which is to make prediction given known past statuses).

- Unlike conventional embedding-based LSTM RNNs, the proposed models explicitly capture 2-way and 3-way interactions of embeddings.
- It is the first work to use bi-directional LSTM in user behavior prediction. To the best of our knowledge, bi-directional LSTM is only applied in handwriting/speech recognition [30] and bio-informatics [31].
- It shows how the proposed models can be used in the ad viewability problem, which is a highly significant research question in the field of advertising. The experimental results validate the effectiveness of the proposed models using a dataset from a large publisher.

4.3.2 The Real-Life Dataset

Data Description A large web publisher (i.e., Forbes Media) provides user browsing logs collected from real website visits in April 2016 and webpage metadata. The dataset contains 5 million page views. For each page view, it records the user id, page URL, state-level user geo location, user agent, and browsing events, e.g., the user opened/left/read the page. Once a user scrolls to a page depth and stays for 1s, an event is recorded. The page depths whose dwell times are less than 1s will not be recorded in the data, in which case we consider their dwell time to be 0s. The page depth is represented as the percentage of the page, ranging from 1% to 100%.

Each event has a timestamp so that the time that a user spends on a part of a page can be calculated. To infer the current part of a page that a user is looking at, the user log also records the page depths at which the first and the last rows of pixels of the screen are. Thus, we are able to infer the part of the page to which the user scrolls and how long the user stayed at that part of the page. Therefore, the dwell time at a page depth can be calculated from the information provided by the user log. Existing work [40] uses almost the same method to accumulate the dwell time of a viewport position (i.e., the area of a user's browser visible on the screen). However, in this existing work, vertical positions are measured by pixels, instead

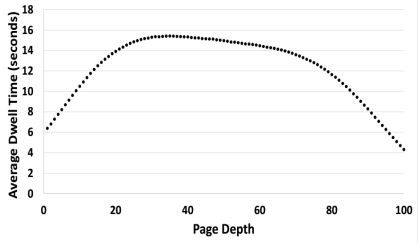


Figure 4.9 The average dwell time of page depths.

of page percentages. The reason that we adopted page percentages is because they provides a relative measure independent of the device screen size. If a user reads 50% of a page on a mobile device, while another user reads 50% of the same page on a desktop, it is assumed that they read the same content.

Empirical Observations We sample 10% of the page views in order to conduct a preliminary data investigation. The average dwell time at each page depth is shown in Figure 4.9. For example, the maximum average dwell time is 15.42s at the page depth of 35%. According to this figure, the average page depth-level dwell time becomes larger initially and then decreases on the second half of webpages. Users spend less time at the top and bottom areas of web pages. This is because the top areas typically contain the navigation bar, mostly titles in big font, or advertisements while the bottom areas contain mostly recommendation to other articles. Users tend to quickly skip these areas and go to the body of content. After reading the body of content, users often leave pages without reading the recommended links.

Figure 4.10 shows the fraction of page depths whose dwell times are at least 1 second, which is the default duration threshold set by IAB. The threshold can be customized by publishers and especially advertisers. In our experiments (Section refsec:experiments), we evaluate the proposed models under different duration thresholds. Figure 4.10 can be derived from Figure 3.2 by setting a dwell time threshold, i.e., 1s because the curves share a very similar shape. It can also

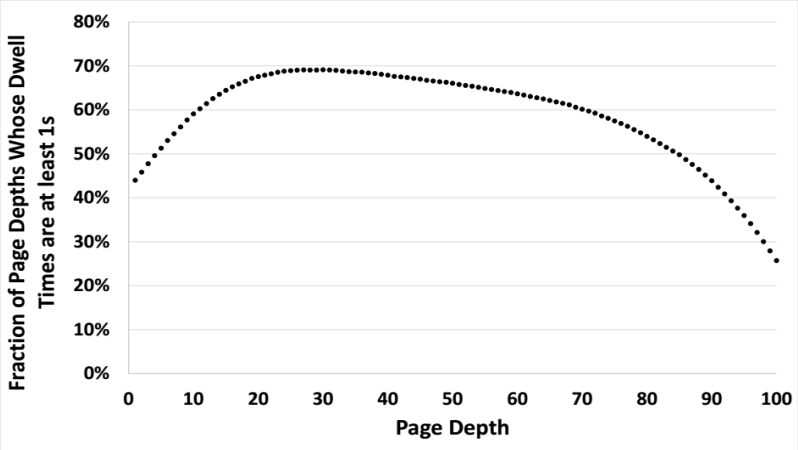


Figure 4.10 The distribution of page views whose dwell times at the corresponding page depths are at least 1s.

be observed that page depth viewability has three phases: It goes up initially and gradually decreases once it reaches the page percentage of 20%. In the last quartile of a page, the viewability goes down at a larger rate.

Figure 4.11 shows the cummulative distribution of page-depth dwell time. In our model, one page view has 100 page depths. As the figure shows, the dwell time of 38.87% page depths is 0 second. This means that the users either quickly scroll past these page depths or leave the pages before scrolling to these page depths. This is intuitive in that users skip the uninteresting areas and only focus on the content they have interest. We also observe that the cumulative percentage for dwell time up to 60 seconds is 94.04%. In other words, users usually spend no more than one minute at one page depth.

Since we use a real-life dataset, it is inevitable to observe outliers in which the depth-level dwell time is extremely high, e.g., hundreds of seconds. This is due to the fact that some users leave the web pages open and go away from their computers. These outliers have no value for the dwell time prediction. Thus, according to the results in Figure 4.11, we set a threshold of 60 seconds for depth-level dwell times.

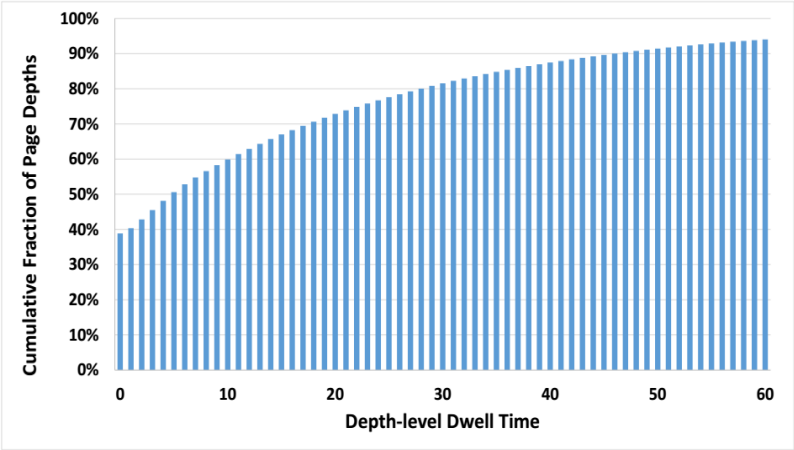


Figure 4.11 The cumulative distribution of page depth dwell time

The entire page view is discarded from the dataset if the dwell time of one of its depth exceeds 60 seconds.

Figure 4.12 shows the distribution of the number of user actions in a page view. A user action is defined as a reading event if a user scrolls to a part of a page and stays for at least one second. A majority of page views have few actions. For example, 26.96% of page views have only 1 action. There are 98.56% page views which have no more than 20 actions. The results make sense because most users do not engage much with pages. It is also observed that outlier page views with as many as 297 user actions exist in the dataset. Therefore, to remove the outliers, we discard the page views that have more than 20 user actions.

4.3.3 Background of LSTM RNN

Before discussing the proposed solutions, we would like to briefly introduce LSTM RNN first.

A recurrent neural network (RNN) is a type of artificial neural network whose connections form cycles, which enable RNN to handle long-term dependencies problems. Unlike feedforward neural networks, RNN can use their internal memory to process arbitrary sequences of inputs. However, traditional RNNs suffer from the

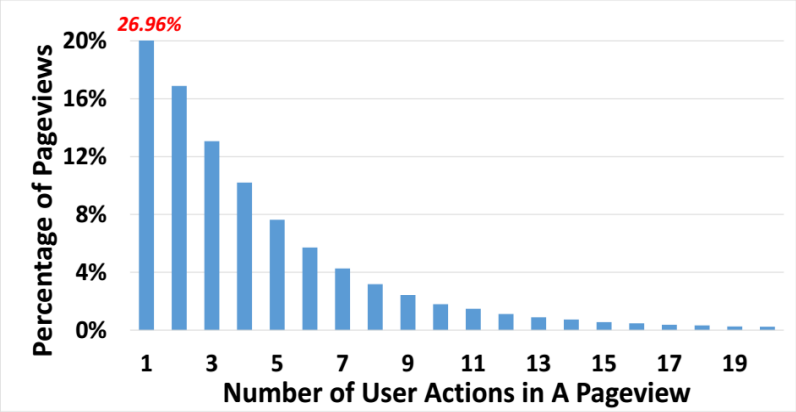


Figure 4.12 The distribution of the number of user actions in a page view.

vanishing or exploding gradient problem [55]: the network output either decays or blows up exponentially as it cycles around the network's recurrent connections, due to the influence of a given input on the hidden layer. Specifically, in the case of decay, the gradient signal between time steps gets smaller so that learning either becomes very slow or stops. This makes the task of learning long-term dependencies in the data more difficult. In addition, if the leading eigenvalue of the weight matrix is more than 1.0, it can increase the gradient signal, so that it can cause learning to diverge.

To avoid the long-term dependency problem, Long Short-term Memory (LSTM) networks [29] have been proposed. The LSTM network is a type of recurrent neural network used in deep learning because it can successfully train for very large architectures. The LSTM networks are good at handling the cases that contain many long sequences. The architecture of LSTM is designed to remember information for long periods of time. The key to LSTMs is the multiplicative gates, which allow LSTM memory cells to store and access information over long periods of time, thereby avoiding the vanishing and exploding gradient problem. Gates are a way to optionally let information through. Researchers use a sigmoid neural net layer and a pointwise multiplication operation to implement gates. The output of the sigmoid neural net layer is either 0 or 1. A value of 0 means a blocked way and a value of 1 means an

unobstructed way. The binary output of the sigmoid network describes how much of each component should be let through. LSTM RNNs have been shown to learn long-term dependencies more easily than the simple RNNs.

The main advantage of LSTM RNN compared to Markov chains and hidden Markov models is that it does not consider the Markov assumption, and thus can be better at exploiting the potential patterns for modeling sequential data. Also, LSTM RNN can discover deep relationship between two time steps, as well as the input of a time step and the outcome. The sequential dependency between the dwell time of different depths is so complex and dynamic that time series analysis of Markov model approaches are not capable to model it effectively. Because of its good performance, LSTM RNN has been used in language modeling [67], speech recognition [30], and user searching behavior [9].

4.3.4 The Proposed LSTM RNN Models

We propose to use LSTM RNN to solve the webpage depth viewability prediction problem. In particular, we developed three models: 1) LSTM RNN; 2) LSTM RNN with embedding interaction; 3) A bi-directional LSTM RNN with embedding interaction.

LSTM RNN Model Understanding and further predicting user engagement on webpages is non-trivial. User engagement is an emotional, cognitive and behavioural connection between a user and a resource, e.g., a webpage [4]. Commonly used metrics to understand user engagement with webpages include scrolling and dwell time. Many factors can influence a user's emotion and cognition when the user is reading a page. It is intuitive that the dwell time of a page depth is highly related to the user's interests and reading habits [22], the topic of the article in the page, aesthetic design at that page depth, etc. Also, content interestingness relates to the emotions experienced during page reading. A viral content may attract most users

to scroll deep on the page and spend a long time on the whole page. Furthermore, different users may have different reading patterns on an interesting page. Finally, aesthetics concerns the sensory and visual appeal of an interface, and it is seen as an important factor for engagement [52]. Webpage layout and graphics may impact the time a user spends at page depths. Page depths with important topic sentences may keep most users longer on them.

Thus, the characteristics of individual users, webpages, and page depths should be taken into account for depth-level dwell time prediction. However, it is non-trivial to explicitly model user interests, page characteristics, and the attractiveness of page depths. Web content publishers usually do not have detailed user profile information, including gender and age. The only user profile they may know is the user agent in the HTTP request and the user geo locations inferred from IP addresses. Also, modeling page interestingness and popularity is still an open research problem. More importantly, the complex interactions of these three factors must be modeled so that their joint effect is captured: 1) The interaction of users and pages captures a user's interest in a page. 2) The interaction of users and page depths can reflect individual users' browsing habits. For example, some users read entire pages carefully, but some only read the upper half. 3) The interaction of pages and depths models the design of individual pages at individual page depths. For example, pages that have a picture at a depth may receive relatively short dwell time at that depth because people usually can understand a picture quicker than text. Therefore, predicting user reading behavior at page depth level is highly challenging. Although implicit feedback, e.g., reading dwell time, is more abundant than explicit feedback, e.g., ratings, it often has higher variability [84], which makes prediction more difficult.

Section 4.2 applies Factorization Machines (FM) to predict the webpage depth-level dwell time. However, the existing solution has two limitations: 1) The latent features learnt by matrix factorization cannot discover the deep joint effect

of the input variables. The FM model learns latent features for input variables through a one-layer shallow network. It has limited ability to discover and utilize deep relationship between the input and the outcome. 2) The user engagement with the previous page depth in the same page view is not considered, despite the fact that it is expected to be a strong indicator for the user behavior at a given page depth. For instance, if a user is predicted to spend long time at the page depths from 1% to 20%, the user probably will stay long at the page depth 21% as well.

This LSTM RNN model addresses these limitations as follows: (1) It uses a deep neural network to capture the underlying patterns between many input factors and webpage depth viewability. (2) The proposed deep learning model takes into account the predicted viewability of the previous page depths in the same page view.

Our LSTM RNN considers the webpage depth-level viewability prediction as a sequential labelling problem, in which the predictions at the time steps (i.e., page depths) can influence the prediction at the current time step. We use LSTM in conjunction with RNN because the length of each sequence in our application is as long as 100 and a traditional RNN will suffer from the vanishing or exploding gradient problem.

Figure 4.13 presents the method used to solve the 'many-to-many' prediction problem by our LSTM RNN. The left side is a webpage, which has 100 page depths. Each page depth corresponds to one time step in the RNN setting, as shown in the right side of the figure. The proposed method makes predictions at every time step. The prediction is the viewability of the page depth in the specific page view. The input of each time step includes information about the user, the page, the depth, and the context. Since the LSTM layers of each time step can generate a viewability prediction, the hidden neurons in the LSTM should carry information about the viewability of that time step. The hidden layers at page depth i should be able to summarize the viewabilities from page depth 1% to i. Therefore, using LSTM to pass

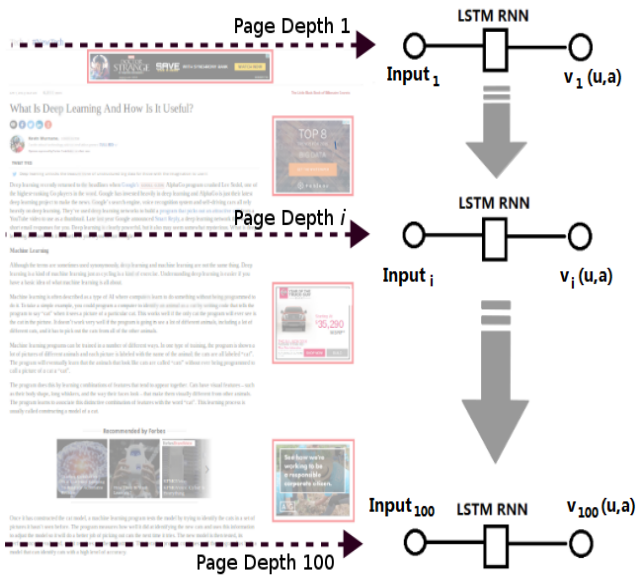


Figure 4.13 Modelling webpage depth viewability prediction.
Adapted from https://www.forbes.com/sites/kevinmurnane/2016/04/01/what-is-deep-learning-and-how-is-it-useful
(accessed on 03/17/2017)

the information of the previous time steps can incorporate the *previously-predicted* viewability into the prediction at the current time step. Note that, since the prediction for a page view is made before page loading, the true viewability of all page depths are unknown. Thus, the *predicted* viewability of the past page depths are used to predict that of the current depth. The outputs at all page depths $v_1(u,a), ..., v_{100}(u,a)$ are counted to compute the performance. Thus, the problem is modeled as a sequence labelling (e.g., Part-of-speech tagging), where the true labels of the past are unknown, instead of time-series (e.g., stock price prediction), where the true labels of the past are used in prediction.

Figure 4.14 presents the architecture of the proposed (rolling) LSTM RNN used in webpage depth viewability prediction. At each page depth, the LSTM RNN consists of one input layer, two LSTM layers, and one output layer.

With the suggestions of the domain experts at Forbes, we consider significant information in the input layer. In particular, the input layer concatenates several components:

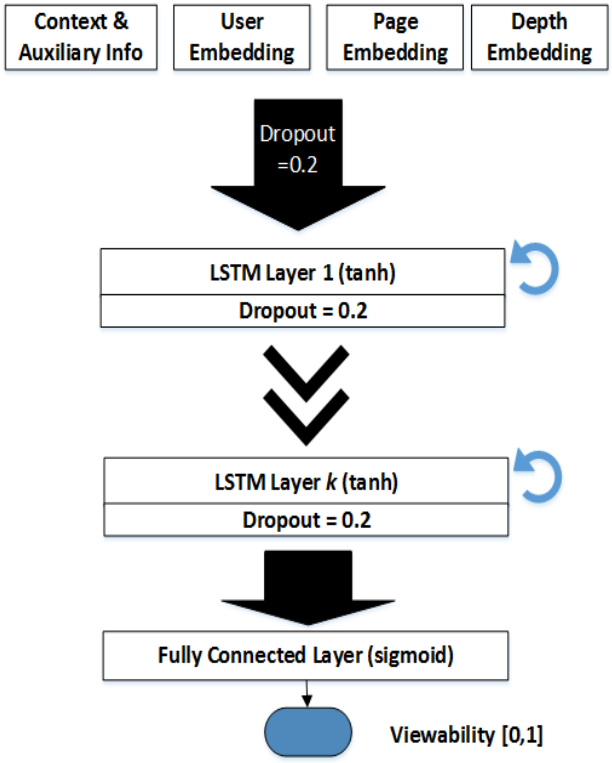


Figure 4.14 The LSTM RNN model.

- The user's *viewport* size, i.e., viewport height and width. A viewport is the part of a user browser visible on the screen. The viewport indicates the device utilized by the user (e.g., a mobile device usually has a much smaller viewport) and can directly determine the user experience. To reduce sparsity, both heights and widths are put into buckets with size 100 pixels.

- The user's *geo location*, which is detected from user IP addresses. Since individual users are identified by cookie IDs, it is possible that the same user visits the website from different locations in multiple sessions. We consider user geo locations because this is the only explicit feature about users that can be easily obtained by publishers without violation of user privacy. In practice, user geo may reflect a user's interests and education. Specifically, geo is the country name if the user is outside USA or a state name if she is within USA.

- *local hour*, and *local day of the week* (denoted by weekday in the experiments), which are likely related to user reading behavior.

- Article *length* is represented by the word count of the article in the page, and it has been proven to be a significant factor impacting page-level dwell time [83]. Article lengths are put into buckets, so that there are a limited number of possible states.

- *Freshness* is the duration between the reading time and the time the page was published on the website. Freshness is measured in days. The freshness of an article may determine the interests of a user on it. Fresh news may receive more user engagement.

- The *channel* and *section* of the article in a page are its topical categories on the publisher's website, e.g., finance and lifestyle. A channel can be considered as a high-level topic label of a page. A section is defined as a sub-channel at finer topical granularity.

- Other page attributes in the Forbes article metadata are also taken into account: page type (e.g., "blog", "blogslide", or "trendingactivity"), whether the page is in standard template type, whether the page contains any image, whether the article is written by Forbes staff, and the number of user comments. All context variables are modeled by one-hot encoding for simplicity. As one common step of feature engineering, rarely-occurred feature values are grouped into "<feature name>_OTHER" categories.

Existing works [9, 91] use mostly one-hot encoding to represent the categorical variables which have millions of values. However, this encoding increases a lot the sparsity and width of the input layers. More importantly, it learns a very limited representation of the variables. Therefore, it is important to use a rich and dense representation to model the most important categorical variables in the data. To this end, our LSTM RNN uses three embedding layers to model the three most important categorical variables: user, page, page depth.

Before concatenating and feeding the four components to the LSTM layers, we apply dropout to each component. Dropout [66] is a simple and effective method to prevent a neural network from overfitting. In particular, it randomly sets a fraction of the units in an output vector to 0 at each update during training time. By cross validation, the fraction is set to 20%.

Stacked above the input layers are multiple LSTM layers. The number of layers is a parameter that can be tuned by experiments. Each can be considered as one reasoning step based on the output of the previous layer. The number of hidden nodes in each LSTM can be empirically determined. Each LSTM layer has two outputs: 1) The first output carries the information of this time step and is sent

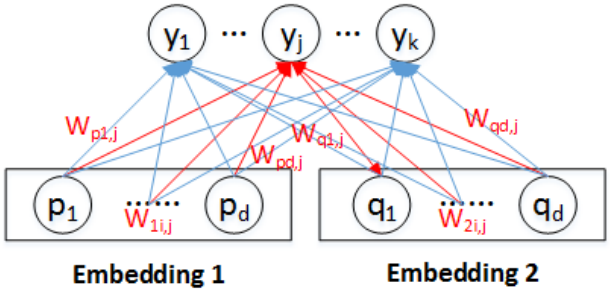


Figure 4.15 Example of propagation without interaction.

to the counterpart at the next time step through a complex set of gates. 2) The second output is passed to the next layer by an activation function. Specifically, the activation function we use is the Tanh function, i.e., $tanh(x) = \frac{2}{1+e^{-2x}} - 1$, which is non-linear and outputs values in the (-1,1) range.

Unlike the sigmoid function whose output is not zero-centered, the Tanh function is less likely to get the network stuck in the current state during training. Also, the Tanh function is not as fragile as the Rectified Linear Unit (ReLU). A large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any data point again.

We also apply dropout to the output of each LSTM layer: 20% units in the output are randomly picked and set to 0. The vertical output of the last LSTM x are passed into a sigmoid function, i.e., $sigmoid(x) = \frac{1}{1+e^{-x}}$, which outputs a value that in the [0,1] range.

The output represents the page depth viewability, i.e., the probability that the page depth will be viewable.

LSTM RNN with Embedding Interactions Model The model presented so far (termed LSTM_noInteract from now on) can be further improved by capturing the interactions of the three important factors: user, page, and page depth. The extended model introduced in this section is denoted as *LSTM_Interact*.

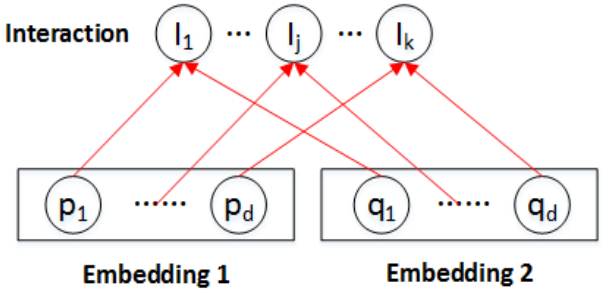


Figure 4.16 Example of propagation with interaction.

Like most general neural networks, the LSTM_noInteract only captures the *OR* relationships among input factors, rather than the *AND* relationship. The input of an activation function at a layer is a linear combination of the input units passed from the previous layer. For instance, Figure 4.15 shows a part of an example network. The two embedding vectors have values $[p_1, p_2, ..., p_d]$ and $[q_1, q_2, ..., q_d]$, respectively. d is the length of the embedding layers. The input of the neuron y_j in the next layer is $\beta_j = \sum_{i=1}^{d} w_{pi,j} b_i + \sum_{i=1}^{d} w_{qi,j} b_i$. The output of y_j is $tanh(\beta_j)$ (assume the activation function adopted is TanH). Thus, like all vanilla neural networks, the LSTM_noInteract does not consider the pairwise interaction of the embedding layers.

To solve this problem, we use knowledge from the recommender system field. For example, to predict a movie rating for an unseen user and movie pair, one simple way is to use matrix factorization, e.g., Singular Vector Decomposition (SVD). SVD learns a latent vector for each user/item. Given an unseen user and item pair, the dot product, i.e., the interaction, of the user latent vector p_u and the page latent vector q_i is the predicted outcome, e.g., a movie rating: $\hat{r}_{ui} = q_i^T p_u$. In other words, it sums up all values generated by element-wise vector multiplication (i.e., multiplying two embedding vectors element by element).

In our case, the embedding vector of an entity can be regarded as a latent vector in SVD. Thus, we can use a similar method to capture the interaction of multiple input factors. In particular, we adopt element-wise embedding multiplication as shown in

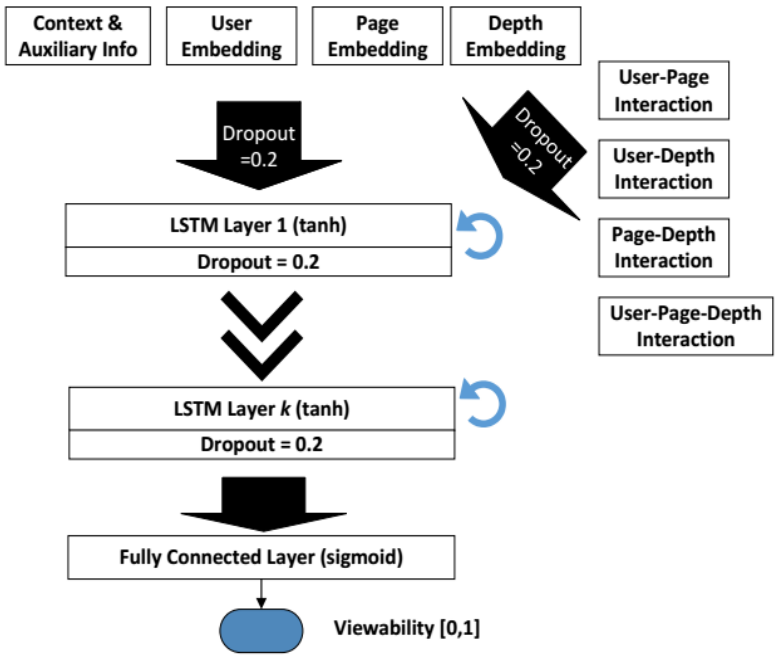


Figure 4.17 The LSTM RNN with embedding interaction model.

Figure 4.16. The ith element y_i in the interaction vector is equal to $y_i = p_i * q_i$. For example, the interaction of [1 0 3] and [2 3 7] is [2 0 21]. The interaction of two embedding layers of length d is a vector of length d. The resulting values will be summed up with other factors in the next layer.

Therefore, at the input layer of the second model, we also consider the 2-way interaction and the 3-way interaction of user, page, and depth embeddings (shown in Figure 4.17). The resulting four interaction vectors are then concatenated with the other input vectors that are already considered in the LSTM_noInteract. This extended model is denoted as *LSTM_Interact*.

Bi-directional LSTM RNN Model In LSTM layers, the state at a time step captures only information from the past and present input. In our application, both LSTM_Interact and LSTM_noInteract proposed so far leverage the sequential dependency among page depths. They predict each output based on the current and previous inputs only: in a same page view, the time that a user spends at the

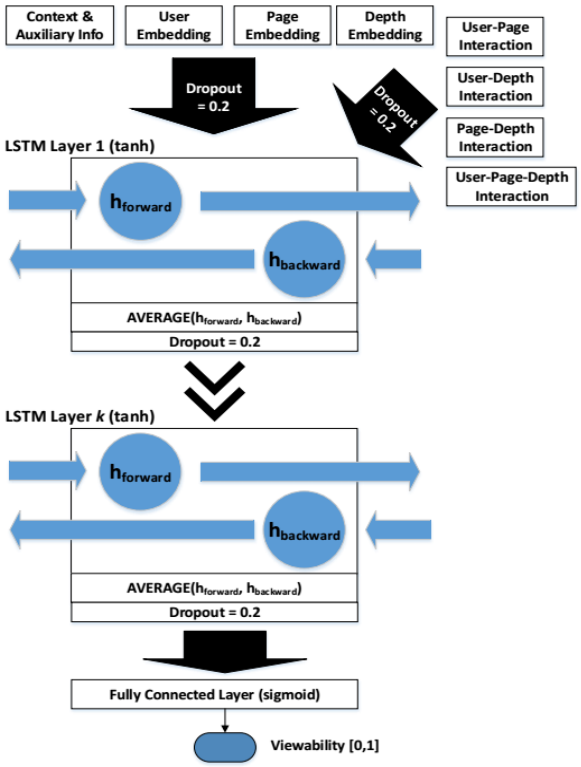


Figure 4.18 The bi-directional LSTM RNN with embedding interaction.

past page depths can indicate the time that the user will spend at the subsequent page depths. Such sequence is formed from the top of a page to the bottom because naturally a page is read in a top-down fashion.

However, it is commonly-observed that users often scroll-back on pages as well. Therefore, dwell time of lower page depths could indicate the dwell time of upper page depths. For instance, a user who spent a long time at the last paragraph of an article and is scrolling up will probably stay long in the middle of the page. The possible reason is that the last paragraph may rekindle the user's interest to the entire article. In this case, single directional LSTMs fail to capture such backward patterns to improve prediction performance.

Moreover, in single directional LSTM, predictions made at very top page depths rely on few previous page depths. For instance, as the Figure 4.13 indicates, only the LSTM layers at the page depth of 1% can contribute to the prediction at the page

depth of 2%. Only the LSTM layers of 1% is known when predicting the output of 2% because predictions are made sequentially from page top to the bottom, i.e., single direction. The information of the page depths after 2% are inaccessible. In this case, relying on very few previous information can lead to unreliable prediction outcome at very top page depths. Less accurate prediction at top page depths may distort the predictions at further page depths. Such problem can be overcome if the page depths after a current one can be taken into account. In this case, the prediction at any page depth considers the information of the other 99 page depths.

Therefore, it may be helpful to enhance the proposed models using a bi-directional LSTM RNN [30], which propagates information in both directions. Bi-directional LSTMs combine a single directional LSTM that moves forward through time, beginning from the start of a sequence, with another LSTM that moves backward through time, beginning from the end of a sequence. It allows the output at a time step t to compute a representation that depends on both the past and the future but in most sensitive to the input around the input at t, without having to specify a fixed-size window around t. Bi-directional LSTMs are useful in some applications, where the prediction outcome of a time step depends on the entire input sequence, rather than only the past and present.

In our case, the bi-directional LSTMs replace the LSTM layers in Figure 4.17: one forward LSTM and one backward LSTM running in reverse direction and with their outputs merged at the output layer. Figure 4.18 shows the architecture of the proposed bi-directional LSTM RNN. The forward LSTM operates as usual: it carries information of the past page depths. In contrast, the backward LSTM flows from the page bottom to the top. It brings information about the dwell time of the page depths that are lower than the current page depth. Along with the input at the current page depth, their outputs are then merged by element-wise averaging because they would equally contribute to the outcome of the page depth. A dropout of 20% follows to

avoid overfitting. In this way, bi-directional LSTMs enable information from both past and future to come together.

Bi-directional LSTM RNNs have been used in speech recognition [30] and bio-informatics [31]. To the best of our knowledge, we are the first to apply bi-directional LSTM networks in user behavior prediction. This bi-directional LSTM RNN is denoted as *Bi-LSTM_Interact*.

4.3.5 Evaluation

Experimental Datasets A three-week user log is split into training and testing data. The training data and testing data contain 1M+ page views and 50K+ page views, respectively. The training/test data consist of all depths of all training/test page views. Note that LSTM RNN parameters are shared by all time steps (i.e., page depths), and each page view contains 100 depths.

At the beginning of each epoch, the training data is shuffled and further split into a new training set and a validation set. The error on the validation set is used as a proxy for the generalization error. We use validation-based early stopping to obtain the models that work the best with the validation data. Since it is common that the validation error may fluctuate during training (producing multiple local minima), the maximum number of epochs is set to 30. By observing the curve of validation errors, it can be guaranteed that overfitting occurs within the first 30 epochs. The models with the minimum validation error are saved and used to predict the testing data. We observe that the minimum validation errors are often obtained at the 8th-15th epochs. In addition, since the exact prediction performance varies by several factors, e.g., parameter initialization, all models are run three times. The reported performance results are obtained by averaging the three runs.

Implementation The proposed LSTM RNN models are implemented using Keras [21] with Theano [70] backend. The experiments are run on a desktop with i7 3.60Hz CPU

and 32GB RAM. The matrix computation is sped up using NVIDA GeForce GTX 1060 6G GPU. Running 30 epochs usually takes 5-8 hours depends on the parameter setting.

Considering the training speed and memory consumption, we set the training batch size to 256. A large batch size might alleviate the impact of noisy data, while a small one sometimes can accelerate the convergence. Hence, we varied the batch size, e.g., 128 and 512, but no significant differences have been observed.

During training for viewability prediction, we use the log-loss function as the objective function, i.e., binary crossentropy, the preferred loss function for binary classification problems. The parameter is initialized by sampling from a uniform distribution. The optimizer we adopt is Stochastic Gradient Descent (SGD), which can can overcome the high cost of backpropagation and still lead to fast convergence, with a learning rate of 0.01, a learning rate decay of 1e-6, and a momentum of 0.99. Nesterov momentum is also enabled. An existing study [68] finds that momentum-accelerated SGD are effective for training RNNs. We also tried RMSprop and Adam optimizers. Although Adam can further accelerate convergence, neither beats SGD for prediction performance.

Comparison Models **GlobalAverage:** In a training page view, if the dwell time of a page depth is at least t seconds, the viewability of the page depth in the page view is considered to be 1; otherwise, it is 0. Therefore, we can calculate what percentage of page views are viewable at each page depth. The 100 constant numbers obtained are used to make deterministic predictions for the corresponding test page depths.

Logistic Regression (LR): Since the viewability prediction can be considered as a classification problem. A logistic regression model is developed as a baseline. The input variables are almost the same as those used in the proposed model. LR models user, page, and depth using one-hot encoding. We develop a one-layer neural network

with sigmoid activation function to mimic a logistic regression. The LR is trained following the same process as the proposed models. The learning method is also SGD with learning rate 0.001.

Factorization Machines (FM): Factorization machines (FM) [60] are a generic approach that combines the high-prediction accuracy of factorization models with the flexibility of feature engineering. This method has been widely used in many applications. The basic idea of FM is to model each target variable as a linear combination of interactions between input variables. Formally, it is defined as: $\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \langle \mathbf{v_i}, \mathbf{v_j} \rangle x_i x_j$, where, $\hat{y}(\mathbf{x})$ is the prediction outcome given an input $\mathbf{x}$. w_0 is a global bias, i.e., the overall average depth-level dwell time. $\sum_{i=1}^{n} w_i x_i$ is the bias of individual input variables. For example, some users would like to read more carefully than others; some pages can attract users to spend more time on them; some page depths, e.g., very bottom of a page, usually receive little dwell time. The first two terms are the same as in linear regression. The third term captures the sparse interaction between each pair of input variables. The FM model is implemented based on [76]. The proposed FM model considers four input variables: user, page, depth, and viewport size.

Metrics The metrics we adopt focus on different aspects of the effectiveness:

Logistic Loss: It is widely used in probabilistic classification. It penalizes a method more for being both confident and wrong. Lower values are better: $logloss = -\frac{1}{N \cdot 100} \sum_{i=1}^{N} \sum_{j=1}^{100} [y_{ij} \log(\hat{y}_{ij}) + (1 - y_{ij}) \log(1 - \hat{y}_{ij})]$, where N is the number of the test page views. Each has 100 page depths, i.e., 100 prediction outputs. $\hat{y}_{ij}$ is the predicted viewability and y_{ij} is the actual viewability at the jth page depth in the ith page view.

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.

Accuracy: The accuracy classification score computes the percentage of the test instances which are correctly predicted. In the experiments, accuracy is computed by the default decision boundary, 0.5. Higher values are better.

Root-mean-square Deviation (RMSD): It is used in dwell time prediction, which is a regression problem. RMSD measures the differences between the values predicted, $\hat{y}_i$, and the values observed, y_i: $RMSD = \sqrt{\frac{\sum_{j=1}^{N}\sum_{i=1}^{100}(\hat{y}_{ij}-y_{ij})^2}{N \cdot 100}}$, where N is the number of test page views. y_{ij} is the actual dwell time at the jth page depth in the ith page view. Lower values are better.

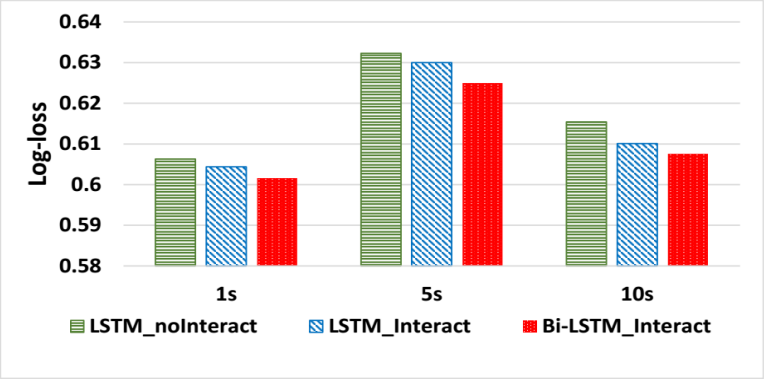


Figure 4.19 Log-loss performance of the proposed models

Comparison of The Proposed Models This section compares the performance of the proposed models. Figure 4.19 shows their performance in the test data. The models contains two LSTM layers, each of which has 500 hidden neurons. The embedding layers have 500 hidden neurons. This network configuration is obtained

experimentally, as discussed in Section 4.3.5. To evaluate the models' performance with different viewability thresholds, i.e., minimum required dwell time, we set three thresholds: 1s, 5s, and 10s. The viewability threshold of 1s is in line with the viewability definition suggested by the IAB. In the experiment, the models are compared by log-loss, accuracy and AUC. However, since we observed that the model with lower log-loss also has higher accuracy and AUC, only the log-loss results are shown.

The Bi-LSTM_Interact model performs best, as it leverages the predicted behaviors at both past page depths and future page depths. The results verify that such bi-directional patterns can consistently enhance the performance by all metrics under all three viewability thresholds. We also notice that LSTM_Interact performs better than LSTM_noInteract because it captures the embedding interaction, which can further boost the prediction performance

Comparing across viewability thresholds, the performance for 1s is the best. Surprisingly, the performance for 5s is not as high as that of 10s. This is because the number of positive instances, i.e., the page depths whose dwell times are at least 5s, and negative instances, i.e., the page depths whose dwell times are less than 5s are almost equal (Figure 4.11). This makes the prediction more challenging.

Table 4.5 Viewability Prediction. Threshold = 1s

Approaches	**Logloss**	**Accuracy**	**AUC**
GlobalAverage	0.6586	59.45%	61.98%
Logistic Regression	0.6484	63.36%	62.55%
FM	0.6173	64.88%	66.11%
Bi-LSTM_Interact	**0.5916**	**67.36**%	**71.54**%

Table 4.6 Viewability Prediction. Threshold = 5s

Approaches	**Logloss**	**Accuracy**	**AUC**
GlobalAverage	0.6786	57.35%	59.42%
Logistic Regression	0.6726	58.94%	61.47%
FM	0.6527	60.13%	63.54%
Bi-LSTM_Interact	**0.6250**	**64.87**%	**70.72**%

Table 4.7 Viewability Prediction. Threshold = 10s

Approaches	**Logloss**	**Accuracy**	**AUC**
GlobalAverage	0.6635	59.65%	58.46%
Logistic Regression	0.6546	60.40%	61.61%
FM	0.6332	62.58%	63.96%
Bi-LSTM_Interact	**0.6076**	**66.98**%	**70.94**%

Performance of Viewability Prediction at All Page Depths This section compares our best model, Bi-LSTM_Interact, and the comparison methods using the log-loss, accuracy, and AUC metrics. The parameter setting of the proposed model is the same as the one used previously.

Tables 4.5, 4.6, and 4.7 present the performance comparison with three viewability thresholds. The Bi-LSTM_Interact model is clearly better than the comparison methods for all three metrics: The lower log-loss indicates that our model has fewer mistakes due to over-confidence when the decision boundary is 0.5. This is also reflected by the accuracy results. The higher AUC values show that our model also obtains better performance on average by varying the decision boundary from 0 to 1. The results verify that bi-directional patterns can consistently enhance the performance for all metrics under all three thresholds.

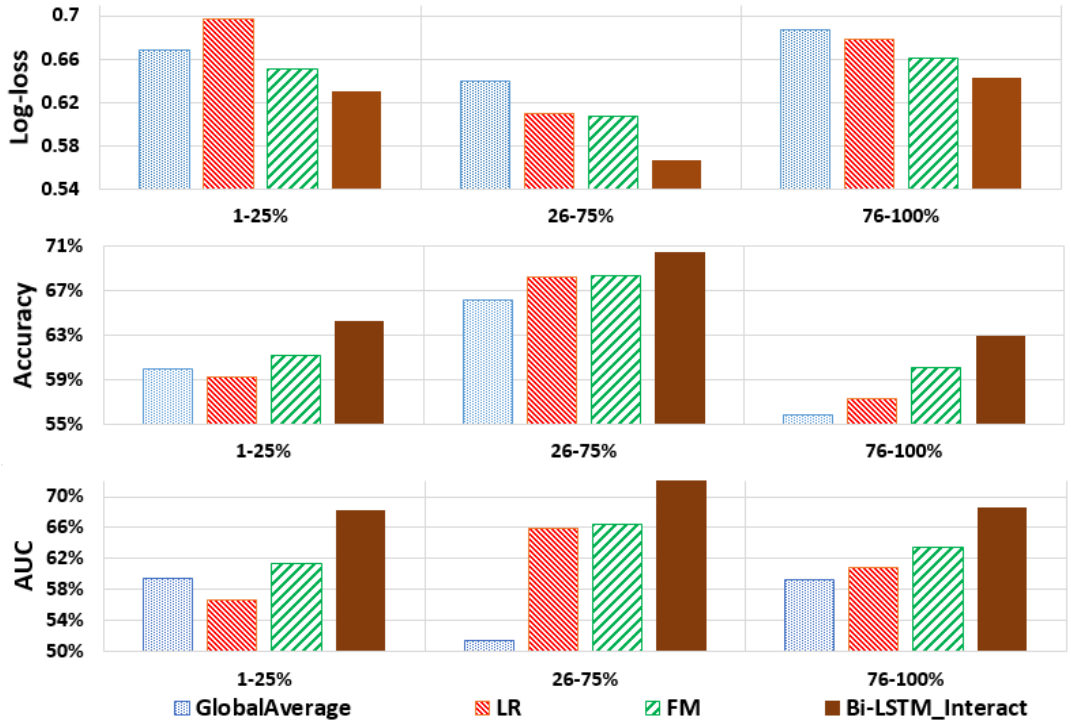


Figure 4.20 Performance of viewability prediction in buckets.

The performance of Bi-LSTM_Interact is significantly better than that of FM. This shows that a deep neural network can discover more underlying patterns between input variables. These patterns can improve the overall performance. The dependency between page depths can also contribute to viewability prediction. The Logistic Regression uses one-hot encoding to represent the user, page, and page depths. Thus, it has limited capability to fit the data as well as the FM which builds latent vectors to model the three categorical variables. Using latent vectors instead of one single weight to model an input variable can increase model complexity. Models with higher complexity can fit datasets better. The GlobalAverage always makes the deterministic prediction using the fraction of positive page depths computed based on the training data. Its performance is the lowest compared with the other methods.

To further evaluate the performance of all methods at different areas of pages, page depths are separated into one of three buckets based on the observations from Figure 4.10: [1,25%], [26%,75%], or [76%,100%]. For brevity, we only present the

results with viewability threshold of 1s because it is the suggested one by the IAB. Figure 4.20 shows that Bi-LSTM_Interact consistently outperforms the comparison systems in all page depth buckets. It is interesting to notice that the performance is better in the middle of the pages, where the log-loss scores are significantly lower and the accuracy and AUC scores are significantly higher. This can be explained by looking at Figure 4.10, which shows more than 50% page views that have middle parts in-view for at least 1s. The reason is that users tend to spend more time on the article content. Thus, it is relatively easier to predict viewability in the middle area.

Comparing the performance at the top and the bottom areas, we can see that the performance at the bottom is clearly worse than that at the top for accuracy. On the other hand, the AUC performance at the bottom is slightly better than that at the top. Since accuracy is calculated using 0.5 as the decision boundary, this indicates that 0.5 does not appear to be a suitable decision boundary at the page bottom. The reason is that the dwell times of most bottom page depths are lower than 1s. Such imbalance leads the probabilistic outputs of all methods to be closer to 0, instead of 0.5; using 0.5 as the decision boundary makes a large majority of predictions 0. Thus, varying the decision threshold, especially lowering it in this case, can enhance the prediction performance.

Effect of Main Parameters In this section, we tune the model by varying some important parameters: the sizes of the embedding layers, the size of the LSTM layers, and the number of LSTM layers. In these experiments, the minimum dwell time threshold is set to 5s, in which case the number of negative training instances is almost the same as the number of positive training cases. Note that the same experiments are conducted for the other two proposed models, but due space constraints, we show results only for the Bi-LSTM_Interact that achieves the best performance.

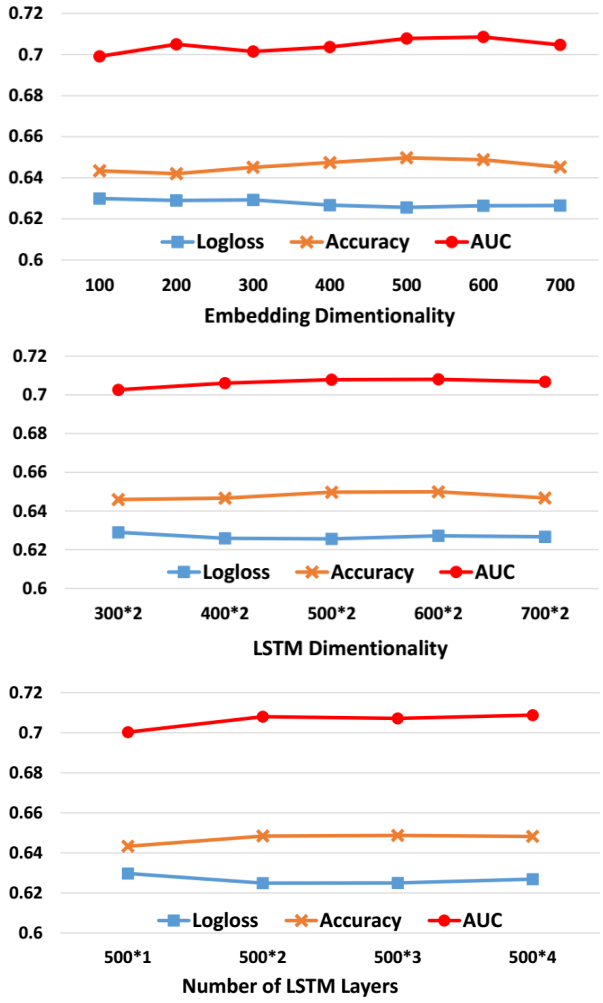


Figure 4.21 Effect of main parameters.

Figure 4.21(a) shows the effect of the dimensionality of the embedding layers, i.e., the number of hidden neurons in an embedding layer. The metrics are computed over the test data. To simplify the solution space, we apply the same dimensionality for user embedding, page embedding, and depth embedding. Varying the dimensionality of the embedding layers also change the dimensionality of the interactions. When varying the embedding layer sizes, we fix the dimentionalities of the bi-directional LSTM layers to 500.

The results show that higher-dimensional word embeddings do not always provide better performance. This finding is in line with existing work [71] that applies word embeddings in a Name Entity Recognition task. Although intuitively wider embedding layers should have finer representation, they also tends to cause overfitting. On the other hand, too narrow embedding layers cannot capture well the traits of input variables.

Embedding=500 obtains the lowest log-loss in the test data. It also has the highest accuracy. However, Embedding=600 is slightly better than Embedding=500 for the AUC score. This implies that Embedding=500 is better than Embedding=600 for the decision threshold of 0.5. But when considering all thresholds on average, Embedding=600 is slightly better. On the other hand, narrower embedding layers have fewer parameters to learn, which requires less training data and shorter training time. Therefore, Embedding=500 may still be preferable in practice.

Figure 4.21(b) shows the effect of the dimensionality of the bi-directional LSTM layers. The number of layers is fixed to two. The results demonstrate that two bi-directional LSTMs with 500 hidden neurons have the best performance for log-loss and accuracy, while the two bi-directional LSTMs with 600 hidden layers lead to the highest AUC by 0.02%. Considering Figure 4.21(a), it appears that hidden layers with 600 hidden neurons are likely to have better performance for all thresholds on

average. However, 500 is better when the decision threshold is 0.5, which is usually the default value.

Figure 4.21(c) presents the effect of the numbers of the bi-directional LSTM layers stacked sequentially. It clearly illustrates that the performance can be significantly improved by increasing the number of bi-directional LSTMs from 1 to 2. But all three performance curves become flat for 3 layers and worse after adding the fourth layer.

Therefore, no one single parameter setting can dominate under all metrics. These experiments in fact reflect two commonly used methods of deep network tuning: go deeper and go wider. Theoretically, a wider deep network can learn a richer representation of the input entity: A hidden layer with more hidden nodes can capture more latent features of a user, a page, a depth, or their interaction with the context. On the other hand, a deeper network that has more hidden layers should be able to learn complex logic processes. In this work, the proposed models require relatively wide embedding layers and bi-directional LSTM layers. The reason is that it is necessary to capture more latent aspects of individual user, page, and depth without explicit features. In addition, Figure 4.21(c) shows that two LSTM layers are enough to fit well the user behavior data. Although user behavior is difficult to learn, the user behavior prediction problem usually does not require too many reasoning steps vertically at each time step [9], compared with the deep networks used in computer vision field [65].

Performance of Dwell Time Prediction The proposed models for page depth viewability prediction can also be applied to predict the exact dwell time of a page depth. This can also be useful for publishers and advertisers: For example, predicted depth-level dwell time can help publishers quantify the interestingness of a page depth to a specific user. In this case, the publisher can determine at which depth is preferable

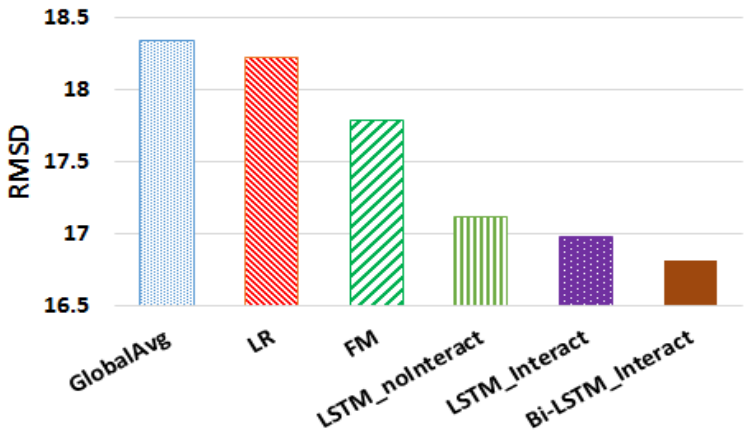


Figure 4.22 Performance of dwell time prediction.

to show recommended article links. Also, advertisers may want to know the exact dwell time that a user will spend at a page depth.

The main difference between viewability prediction and dwell time prediction is the output. The former, a classification problem, outputs a probability, i.e., [0,1]; the latter, a regression problem, outputs a non-negative value, i.e., [0,+∞] (time can not be negative). Thus, in order to make page depth dwell time prediction, we change the activation function of the output layer from a sigmoid function to a rectified linear unit (ReLU). Given a input x of a linear combination sent from the previous layer, ReLU converts it by $relu(x) = max(0, x)$. Thus, the output of a ReLU is a non-negative value. In addition, the learning rate is reduced from 0.01 to 0.001 because 0.01 is too large for the regression problem. We use the mean squared error as the objective function. The results are calculated based on all test page depths, instead of buckets. Figure 4.22 shows that the all three proposed models significantly outperform the comparison systems. The Bi-LSTM_Interact generates the least RMSD.

4.4 Chapter Conclusion

Online publishers and advertisers are interested to predict how likely it is that a user will stay at a page depth for at least a certain dwell time, defined as webpage

depth viewability. Viewability prediction can maximize publishers' ad revenue and boost advertisers' return on investment. This work first proposes a model based on Factorization Machines to predict webpage depth-level viewability for a page view. Using real-world data, both page depth-level dwell time and viewability prediction experiments consistently show the proposed FM model outperforms the comparison models. This work then proposes three deep sequential neural networks based on Recurrent Neural Network (RNN) with the Long Short-Term Memory (LSTM). The proposed deep learning models predict the viewability and exact dwell time for any page depth in a specific page view. Using a real-world dataset, the experiments consistently show our models outperforming the comparison models.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The aim of this dissertation is to advance the state-of-the-art research on ad viewability prediction. As an emerging research topic, ad viewability obtains its significance in industry. Ad viewability prediction can improve the effects and efficiency of advertisers' marketing investment and, on the other side of the table, increase publishers advertising revenue. As a result, more efficient advertising campaigns enhance advertisers profit. More ad revenue help small and medium publishers sustain from intense business competition.

According to the standard definition of a viewable impression, ad viewability is proposed to be predicted from two different angles: by scrolling behavior and dwell time. The first angle leverages the fact that users often do not scroll down enough to make an ad impression shown on screen. Two machine learning models are proposed based on the probabilistic latent class models: PLC_const and PLC_dyn. The first model computes constant memberships of each user and page offline, while the second model computes dynamic memberships in the real time. The experimental results show that the proposed models outperform the state-of-the-art methods. However, predicting ad viewability by scrolling behavior does not take into account the time that an ad is shown on the screen. Ad dwell time is also a requirement in the standard definition of ad viewability. Thus, the second phase proposes to estimate ad viewability by predicting dwell time of the page depth where an ad is located. Factorization machines models with different feature combinations are first proposed. The one with Doc2Vec vectors, viewport, and channel obtains the best prediction performance. To discover and leverage the deep patterns among input variables, three deep sequential neural networks are then proposed. In particular, three RNN

LSTM models are built to improve the ad viewability prediction performance. The experiments shows that the deep learning models outperform the comparison models. The bidirectional LSTM model performs the best.

All proposed models predict ad viewability before a user reads a page. The LSTM models have the best prediction performance. They need more input training data and offline training time in order to learn the optimal values of parameters. In contrast, the PLC_dyn need the least training data. This is preferable for those cases where huge training data are not available. In addition, the FM-based models are highly suitable for industry because they are easy to be implemented and customized.

The contribution of this research includes: 1) We are the first to define and solve the problem of viewability prediction, which is a significant problem in online advertising. 2) We are also the first to predict user scrolling behavior. 3) We are the first to investigate the depth-level dwell time prediction by developing machines learning models. 4) The proposed models are evaluated using real-life datasets. The experimental results show that our models significantly outperform the comparison systems.

There are several potential future research directions. Since users may have different behaviors on different devices. We plan to differentiate devices (i.e., desktop or mobile) and browsers. The performance of the proposed models on different devices and browsers will be evaluated and compared. Also, the hierarchical relationship among page, section, and channel, as well as user and geo-location can be further utilized. It is promising to investigate whether such relationships can contribute the ad viewability prediction. Another direction is to predict ad viewability during page reading. In addition, some publishers may prefer to dynamically present ads during page reading. An ad impression is not determined and sold until a user scrolls to the page depth where the ad is located. Thus, the ad viewability can be largely increases.

It is helpful to estimate the viewability of an ad located at a deeper page depth when a user is reading a page.

BIBLIOGRAPHY

[1] Deepak Agarwal, Bo Long, Jonathan Traupman, Doris Xin, and Liang Zhang. Laser: A scalable response prediction platform for online advertising. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 173–182, 2014.

[2] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 19–26, 2006.

[3] Mohamed Aly, Sandeep Pandey, Vanja Josifovski, and Kunal Punera. Towards a robust modeling of temporal interest change patterns for behavioral targeting. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 71–82, 2013.

[4] Simon Attfield, Gabriella Kazai, Mounia Lalmas, and Benjamin Piwowarski. Towards a science of user engagement. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining Workshop*, 2011.

[5] Santiago R Balseiro, Jon Feldman, Vahab Mirrokni, and S Muthukrishnan. Yield optimization of display advertising with ad exchange. *Management Science*, 60(12):2886–2907, 2014.

[6] MohammadHossein Bateni, Jon Feldman, Vahab Mirrokni, and Sam Chiu-wai Wong. Multiplicative bidding in online advertising. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 715–732. ACM, 2014.

[7] Mikhail Bilenko and Matthew Richardson. Predictive client-side profiles for personalized advertising. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 413–421, 2011.

[8] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[9] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. A neural click model for web search. In *WWW*, pages 531–541, 2016.

[10] Davidson-Pilon C. Lifelines. https://github.com/camdavidsonpilon/lifelines, 2016.

[11] Suleyman Cetintas, Datong Chen, and Luo Si. Forecasting user visits for online display advertising. *Information Retrieval*, 16(3):369–390, 2013.

[12] Suleyman Cetintas, Luo Si, Yan Ping Xin, and Ron Tzur. Probabilistic latent class models for predicting student performance. In *Proceedings of the 22nd ACM*

CIKM Conference on Information and Knowledge Management, pages 1513–1516. ACM, 2013.

[13] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transaction on Intelligent Systems Technology*, 5(4):61:1–61:34, December 2014.

[14] Jianqing Chen and Jan Stallaert. An economic analysis of online advertising using behavioral targeting. *MIS Q.*, 38(2):429–450, June 2014.

[15] Wei Chen, Di He, Tie-Yan Liu, Tao Qin, Yixin Tao, and Liwei Wang. Generalized second price auction with probabilistic broad match. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 39–56. ACM, 2014.

[16] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1307–1315. ACM, 2011.

[17] Ye Chen, Dmitry Pavlov, and John F. Canny. Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 209–218, 2009.

[18] Ye Chen and Tak W Yan. Position-normalized click prediction in search advertising. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 795–803. ACM, 2012.

[19] Ying-Ju Chen. Optimal dynamic auctions for display advertising. *Available at SSRN 2216361*, 2013.

[20] Haibin Cheng, Eren Manavoglu, Ying Cui, Ruofei Zhang, and Jianchang 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*, ADKDD '12, pages 9:1–9:9, 2012.

[21] Franois Chollet. keras. https://github.com/fchollet/keras, 2015.

[22] Mark Claypool, Phong Le, Makoto Wased, and David Brown. Implicit interest indicators. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*, IUI '01, pages 33–40, 2001.

[23] Greg Corrado. Deep networks for predicting ad click through rates. In *International Conference on Machine Learning Online Advertising Workshop*, 2012.

[24] Yanqing Cui and Virpi Roto. How people use the web on mobile devices. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 905–914, 2008.

[25] Jon Feldman, S Muthukrishnan, Martin Pal, and Cliff Stein. Budget optimization in search-based advertising auctions. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 40–49. ACM, 2007.

[26] Stephanie Flosi, Gian Fulgoni, and Adrea Vollman. If an advertisement runs online and no one sees it, is it still an ad? *Journal of Advertising Research*, 2013.

[27] Google. The importance of being seen. https://think.storage.googleapis.com/docs/the-importance-of-being-seen_study.pdf (accessed on 03/17/2017), 2014.

[28] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 13–20, 2010.

[29] Alex Graves. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer, 2012.

[30] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

[31] Jack Hanson, Yuedong Yang, Kuldip Paliwal, and Yaoqi Zhou. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks. *Bioinformatics*, 33(5):685, 2017.

[32] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

[33] Michal Holub and Maria Bielikova. Estimation of user interest in visited web page. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1111–1112, 2010.

[34] Liangjie Hong, Aziz S. Doumith, and Brian D. Davison. Co-factorization machines: Modeling user interests and predicting individual decisions in twitter. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 557–566, 2013.

[35] Bernard J. Jansen. Click fraud. *Computer*, 40(7):85–86, July 2007.

[36] Youngho Kim, Ahmed Hassan, Ryen W. White, and Imed Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 193–202, 2014.

[37] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[38] Subodha Kumar and Suresh P Sethi. Dynamic pricing and advertising for web content providers. *European Journal of Operational Research*, 197(3):924–944, 2009.

[39] S Kyle. Experimenting in loyalty conversion with wnyc: Achieving mobile-desktop parity. http://blog.chartbeat.com/2013/10/07/experimenting-loyalty-conversion-wnyc-achieving-mobile-desktop-parity/.

[40] Dmitry Lagun and Mounia Lalmas. Understanding user attention and engagement in online news reading. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 113–122. ACM, 2016.

[41] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[42] Chao Liu, Ryen W. White, and Susan Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 379–386, 2010.

[43] Kun Liu and Lei Tang. Large-scale behavioral targeting with a social twist. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1815–1824, 2011.

[44] Ingrid 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/, 2014.

[45] F. Manjoo. You wont finish this article. http://www.slate.com/articles/technology/technology/2013/06/how_people_read_online_why_you_won_t_finish_this_article.html, 2013.

[46] Manfred Mareck. Is online audience measurement coming of age? *Research World*, 2015(51):16–19, 2015.

[47] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1222–1230. ACM, 2013.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, 2013.

[49] Ananth Mohan, Zheng Chen, and Kilian Q Weinberger. Web-search ranking with initialized gradient boosted regression trees. In *Yahoo! Learning to Rank Challenge*, pages 77–89, 2011.

[50] Mohamed Mostagir. Optimal delivery in display advertising. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 577–583. IEEE, 2010.

[51] Sami Najafi-Asadolahi and Kristin Fridgeirsdottir. Cost-per-click pricing for display advertising. *Manufacturing & Service Operations Management*, 16(4):482–497, 2014.

[52] Heather L. O'Brien and Elaine G. Toms. The development and evaluation of a survey to measure user engagement. *J. Am. Soc. Inf. Sci. Technol.*, 61(1):50–69, January 2010.

[53] Michael Ostrovsky and Michael Schwarz. Reserve prices in internet advertising auctions: A field experiment. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 59–60. ACM, 2011.

[54] Sandeep Pandey, Mohamed Aly, Abraham Bagherjeiran, Andrew Hatch, Peter Ciccolo, Adwait Ratnaparkhi, and Martin Zinkevich. Learning to target: What works for behavioral targeting. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1805–1814, 2011.

[55] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on International Conference on Machine Learning*, pages 1310–1318, 2013.

[56] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. Bid optimizing and inventory scoring in targeted online advertising. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 804–812. ACM, 2012.

[57] Runwei Qiang, Feng Liang, and Jianwu Yang. Exploiting ranking factorization machines for microblog retrieval. In *Proceedings of the 22nd ACM international Conference on Information and Knowledge Management*, CIKM '13, pages 1783–1788, 2013.

[58] Ana Radovanovic and William D Heavlin. Risk-aware revenue maximization in display advertising. In *Proceedings of the 21st International Conference on World Wide Web*, pages 91–100. ACM, 2012.

[59] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the International Conference on Language Resources and Evaluation Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[60] Steffen Rendle. Factorization machines with libfm. *ACM Transaction on Intelligent System Technology*, 3(3):57:1–57:22, May 2012.

[61] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web*, pages 521–530. ACM, 2007.

[62] Guillaume Roels and Kristin Fridgeirsdottir. Dynamic revenue management for online display advertising. *Journal of Revenue & Pricing Management*, 8(5):452–466, 2009.

[63] Rómer Rosales, Haibin Cheng, and Eren Manavoglu. Post-click conversion modeling and analysis for non-guaranteed delivery display advertising. In *Proceedings of the fifth ACM International Conference on Web search and data mining*, pages 293–302. ACM, 2012.

[64] Lauren Shupp, Robert Ball, Beth Yost, John Booker, and Chris North. Evaluation of viewport size and curvature of large, high-resolution displays. In *Proceedings of Graphics Interface*, GI '06, pages 123–130, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[65] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Proceedings of the International Conference on Learning Representations*, 2015.

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[67] Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. From feedforward to recurrent lstm neural networks for language modeling. *Trans. Audio, Speech and Lang. Proc.*, 23(3):517–529, March 2015.

[68] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. pages III–1139–III–1147, 2013.

[69] Liang Tang, Romer Rosales, Ajit Singh, and Deepak Agarwal. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd ACM international Conference on Information and Knowledge Management*, CIKM '13, pages 1587–1594, 2013.

[70] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, May 2016.

[71] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of The Annual Meeting of The Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[72] William E Walsh, Craig Boutilier, Tuomas Sandholm, Rob Shields, George Nemhauser, and David C Parkes. Automated channel abstraction for advertising auctions. 2009.

[73] Chieh-Jen Wang and Hsin-Hsi Chen. Learning user behaviors for advertisements click prediction. In *The International ACM SIGIR Conference on Research and Development in Information Retrieval Workshop on Internet Advertising*, pages 1–6, 2011.

[74] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, 2011.

[75] Chong Wang, Achir Kalra, Cristian Borcea, and Yi Chen. Revenue-optimized webpage recommendation. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1558–1559. IEEE, 2015.

[76] Chong Wang, Achir Kalra, Cristian Borcea, and Yi Chen. Webpage depth-level dwell time prediction. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1937–1940, 2016.

[77] Jun Wang and Shuai Yuan. Real-time bidding: A new frontier of computational advertising research. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 415–416. ACM, 2015.

[78] Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. Not quite the average: An empirical study of web use. *ACM Transaction on The Web*, 2(1):5:1–5:31, March 2008.

[79] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transaction on Information Systems*, 26(3):13:1–13:37, June 2008.

[80] Xiaohui Wu, Jun Yan, Ning Liu, Shuicheng Yan, Ying Chen, and Zheng Chen. Probabilistic latent semantic user segmentation for behavioral targeted advertising. In *Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising*, pages 10–17. ACM, 2009.

[81] Songhua Xu, Hao Jiang, and Francis C. M. Lau. Mining user dwell time for personalized web search re-ranking. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three*, IJCAI'11, pages 2367–2372. AAAI Press, 2011.

[82] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 261–270, 2009.

[83] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. Beyond clicks: Dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 113–120, 2014.

[84] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. Silence is also evidence: Interpreting dwell time for recommendation from psychological perspective. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 989–997. ACM, 2013.

[85] Shuai Yuan, Ahmad Zainal Abidin, Marc Sloan, and Jun Wang. Internet advertising: An interplay among advertisers, online publishers, ad exchanges and web users. *arXiv preprint arXiv:1206.1754*, 2012.

[86] Shuai Yuan, Jun Wang, Bowei Chen, Peter Mason, and Sam Seljan. An empirical study of reserve price optimisation in real-time bidding. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1897–1906. ACM, 2014.

[87] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. Real-time bidding for online advertising: Measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ADKDD '13, pages 3:1–3:8, 2013.

[88] Yong Yuan, Feiyue Wang, Juanjuan Li, and Rui Qin. A survey on real time bidding advertising. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 418–423. IEEE, 2014.

[89] Weinan Zhang, Shuai Yuan, and Jun Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1077–1086. ACM, 2014.

[90] Weinan Zhang, Shuai Yuan, Jun Wang, and Xuehua Shen. Real-time bidding benchmarking with ipinyou dataset. *arXiv preprint arXiv:1407.7073*, 2014.

[91] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[92] Zeyuan Allen Zhu, Weizhu Chen, Tom Minka, Chenguang Zhu, and Zheng Chen. A novel click model and its applications to online advertising. In *Proceedings of the third ACM International Conference on Web search and data mining*, pages 321–330. ACM, 2010.