Click-through Prediction for Advertising in Twitter Timeline

Cheng Li [1]*, Yue Lu[2], Qiaozhu Mei[1], Dong Wang[2], Sandeep Pandey[2]
[1]School of Information, University of Michigan, Ann Arbor, MI, USA
[2]Twitter Inc, 1355 Market St Suite 900, San Francisco, CA, USA
lichengz@umich.edu, yuelu@twitter.com, qmei@umich.edu, dongw@twitter.com,
spandey@twitter.com

ABSTRACT

We present the problem of click-through prediction for advertising in Twitter timeline, which displays a stream of Tweets from accounts a user choose to follow. Traditional computational advertising usually appears in two forms: *sponsored search* that places ads onto the search result page when a query is issued to a search engine, and *contextual advertising* that places ads onto a regular, usually static Web page. Compared with these two paradigms, placing ads into a Tweet stream is particularly challenging given the nature of the data stream: the context into which an ad can be placed updates dynamically and never replicates. Every ad is therefore placed into a unique context. This makes the information available for training a machine learning model extremely sparse. In this study, we propose a learning-to-rank method which not only addresses the sparsity of training signals but also can be trained and updated online. The proposed method is evaluated using both offline experiments and online A/B tests, which involve very large collections of Twitter data and real Twitter users. Results of the experiments prove the effectiveness and efficiency of our solution, and its superiority over the current production model adopted by Twitter.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation, Measurements

Keywords

Online Advertising; Social Media Stream; Click-through Prediction

1. INTRODUCTION

Users at Twitter could subscribe to other accounts, commonly referred to as followees. These followees continuously produce messages ("Tweets"), corresponding in general to the follower's long term interest. Once posted, Tweets are pushed into the follower's **timeline**, a continuous stream of Tweets from one's followees. To facilitate the consumption of the large amount of real time information, each user's timeline is displayed in a way that new arrivals are presented on the top of the screen, replacing the older ones. When a user refreshes her timeline, only limited number of Tweets are pushed to the user's device. This leads to the concept of **session**, which consists of all the Tweets sent to one user at the same time.

In this paper, we present the problem of click-through prediction for advertising in Twitter timeline, an essential problem in determining ads to be presented and appropriate payments [11]. Stating the problem more specifically, given a user's timeline, the session pushed to this user at a particular time, and a set of ads, we predict the probability that a particular ad will be clicked on if it is displayed at a particular position of this user's timeline. Traditional online advertising usually appears in two forms: *sponsored search* and *contextual advertising*. Sponsored search is designed for web search engines. It is concerned with placing ads onto a search result page of a particular query. In contrast, contextual advertising studies how to display ads on a regular, usually static Web page. Compared with these two traditional paradigms, placing ads into a Twitter user's timeline is particularly challenging given its streamed nature. First, the stream of Tweets are emitted from accounts the user follows, which usually correspond to her long term interest but do not reflect her current status. However, whether the user clicks on an ad or not depends more on her current information need (a.k.a., intent) when the ad is viewed [30]. For example, a user following *@Microsoft* on Twitter is not necessarily looking for a Microsoft product right now. A user who is inquiring about the most recent version of iPhone may not necessarily subscribe to Apple's feeds. Second, every user receives a unique stream of Tweets which update continuously. Compared with sponsored search where an ad can be placed whenever the same query is issued, and to contextual advertising where an ad can be placed whenever a user visits the same Web page, in a Tweet stream few ads are placed in the same session. Moreover, every user has a different timeline which is updated dynamically. This means that we have a unique "page" (i.e., session) for every user at any given time point. As a result, ads inserted at different time points are actually displayed in completely different "pages" (sessions). These factors make it difficult to gather enough user behavioral signals for training a machine learning model and non-trivial to utilize historical clicks on an ad for predicting how likely it will be clicked on in the future. These unique challenges call for a reevaluation of the techniques employed in traditional online advertising scenarios, and for new approaches that could specifically address the unique properties of Tweet streams.

In this paper, We investigate how learning-to-rank and online learning techniques can be utilized and customized to address these

*The work is done while the author was an intern at Twitter Inc.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s). Copyright is held by the owner/author(s).

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.
ACM 978-1-4503-3664-2/15/08.
DOI: http://dx.doi.org/10.1145/2783258.2788582 .

challenges. We examine our solutions by conducting both offline experiments and online experiments (a.k.a., A/B tests), which involve very large collections of Twitter data and real Twitter users. The results prove the efficiency and effectiveness of the proposed method, and its improvement over Twitter's current production model.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 gives an overview of the procedure for advertising in stream. Section 4 describes methods to tackle the challenges, followed by description of infrastructure implementation to conduct online experiments in Section 5. Section 6 describes setup of offline experiments and results, while the report of online experiments is present in Section 7. Finally, we conclude the paper in Section 8.

2. RELATED WORK

The task of click-through prediction for online advertising at Twitter is related to the practice in social media industry, including online advertising in Facebook, LinkedIn, etc. The most relevant work in literature is a recent study of predicting clicks on Facebook ads by He et al. [16]. Their study did not fully address the unique properties of data streams and the challenges of advertising in streams. As a result, they treated the prediction for every ad as a classification problem, a formulation commonly used in conventional online advertising scenarios. We show in this paper that a significant improvement can be obtained by considering the unique properties of data streams and utilizing learning-to-rank techniques. Researchers at Google [1] developed a system that processes training data in a streamed fashion, which was applied to sponsored search. The problem we present is quite different from and arguably much more challenging than theirs, in which an ad is not triggered by a query but by a unique, fast evolving context in the social media stream.

Aside from studies on advertising in streams, we consider three lines of researches that are related to our work: learning to rank, online advertising, and ranking streaming data. Due to the large body of work on learning to rank [21] and online advertising [12], we constrain our focus to predicting click-through rate (CTR) for online advertising through a machine learning approach. That is, we aim at training CTR prediction models using either historical click logs or human annotated examples.

Traditionally, online advertising falls into two categories: sponsored search [18] and contextual advertising [27]. Sponsored search advertising displays advertisements onto the result page of a particular query submitted to a search engine. Google is the first to incorporate click feedback into the model [3]. In order to estimate CTR for new ads, Richardson et al. [28] explored different types of features (ads, terms, and advertisers) to train a logistic regression model. Considering the rapid growth of data volume, Ciaramita et al. [9] employed online learning using the perceptron algorithm. Graepel et al. [15] addressed the same problem by using a scalable Bayesian algorithm.

Contextual advertising refers to ads placement within the content of regular web pages. A large body of work is devoted to learning the relevance of the displayed ads to the page content. Features identifying both semantic and syntactic relationship of words between the Web page and ads are included in [6]. Semantic relationships of words between Web pages and ads are modeled by hidden classes in [26]. Murdock et al. [23] developed a system that uses features derived from statistical machine translation models, aiming to learn a "translation" of vocabularies between ads and target pages. Chakrabarti et al. [8] applied logistic regression to learn the match between ads and Web pages. Bai et al. [2] proposed a supervised version of latent semantic indexing to map the query-document pair to a ranking score. In order to learn the ranking function, genetic programming is adopted by Lacerda et al. [20]. Karimzadehgan et al. [19] proposed a stochastic algorithm to learn a ranking function that directly optimizes IR evaluation metrics, which are usually not differentiable.

Ranking streaming data, such as Tweets, has been widely studied in literature. Duan et al. [10] presented an empirical study on learning to rank of Tweets by considering features like content relevance, account authority and Tweet-specific features such as presence of a URL. Zhang et al. [31] exploited transductive learning, of which the idea is similar to pseudo relevance feedback, to alleviate the problem of few training data in learning to rank. To find out the semantics of Tweet content, Nguyen et al. [24] incorporated topic modeling features into the learning-to-rank procedure. Hong et al. [17] studied ranking LinkedIn social updates by utilizing learning to rank, collaborative filtering, and click-through modeling. Our work is different from these studies as our target is not ranking organic feeds in the stream, but the ads to be inserted into the stream. The organic feeds in the stream instead defines the context of ads. Our study also goes beyond finding the relative order of candidate ads and targets an accurate estimation of click probability.

3. ADVERTISING IN TWITTER TIMELINE

3.1 System overview

We first give an overview of the advertising system designed for Twitter. This system overview could help the readers better understand how advertising in a message stream functions.

The nature of Twitter encourages various forms of advertisements. For instance, advertisers could invite users to follow their Twitter accounts, enhance the popularity of a particular hashtag, and distribute product information via Tweets.[1] Among them, a large proportion of user targeting takes the form of Tweets, which we call *promoted Tweets*. When inserted in user's timeline, promoted Tweets are like regular Tweets: they scroll through the timeline, appear in the timeline just once, and users can engage with the Tweets by a variety of forms. For example, users can click on URLs, retweet, reply to, or favorite the promoted Tweet, just as they do to any other regular Tweets. The only difference is that a user could perform a negative engagement with a promoted Tweet by hitting a "dismiss" button associated with the promoted Tweets.

Figure 1 summarizes the interactions between the client side and the ad server. When a Twitter user refreshes her home timeline, the client side issues an ad display request to the ad server. We refer to this time stamp as *request time*. An initial set of ad candidates are formed according to the information of the user. To decide the winner from these candidates, an auction is run based on two major factors. The first is the bid price, the amount of money advertisers are willing to pay if users engage with their ads. The second factor is the predicted click probability – the target of our prediction task. We call this task the *click-through rate* (**CTR**) prediction, following the literature of online advertising. By clicks, we mean any type of engagements with the Tweets. In this paper, we focus on predicting the probability of any positive engagements, e.g., retweet, reply, and click a URL. Described methods could be easily generalized to prediction of a specific user action, e.g., dismissing a promoted Tweet. Positive engagements are shortened as *engagements* in the rest of the paper if no ambiguity arises. At the end of the auction, there could be zero to K winning ads. No ad will be placed if the system cannot find a good match to the context. Showing ads anyway in this case would hurt user experience in the long run.

[1] https://business.twitter.com/ad-products

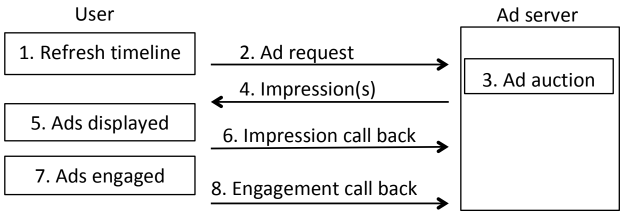


Figure 1: The process of displaying an ad in Twitter timeline. Numbers indicate the order of events.

Due to the same reason, the maximum number of selected ads K is usually set to a very small number.

One thing worth noting is that there are essentially two aspects in the CTR prediction task. On the one hand, we should give a correct estimation of click probability, especially for the winners in the auction. An underestimation could result in no winners while an overestimation incurs user frustration. In addition, inaccurate prediction leads to complications in charging, due to Twitter's adoption of second-price auction [11]. On the other hand, a ranking of the ads are more critical than the actual values of CTR when choosing the best ads to show to every user. This is especially important as there is a very limited number of spots to display ads. A model achieving reasonable CTR estimation does not necessarily output good ranking, and vice versa.

Having chosen the ads to display, the server expects an impression callback from the client, indicating the successful appearance of ads in user's device screen. This time stamp is recorded as *impression callback time*. This callback is necessary given the streamed nature of user's timeline: promoted Tweets might have already been scrolled over before users ever see it. Admittedly, impression callbacks only indicate that promoted Tweets have been displayed in user's screen. It is very possible that the user fails to notice them. However, this is the best signal we can collect that indicates the impression. If, by any chance, the user engages with the promoted Tweet, an engagement callback will be triggered. This point of time is called *engagement callback time*.

4. METHODS

Challenges of CTR prediction at Twitter call for special treatments to the machine learning techniques used in conventional scenarios that could at least address some aspects of these challenges. To this end, our approach in general is to employ learning-to-rank techniques in an online fashion, with careful design of features that addresses the unique properties of data streams. We first introduce a straightforward classification framework, employed as current production model at Twitter, and then investigate possible improvement based on it.

4.1 Pointwise approach

Following the typical machine learning approach to CTR prediction problem, we use a pointwise learning-to-rank approach as our baseline, a method applied as the production model of Twitter. In our particular application, we try to train a probabilistic classifier that assigns a posterior click-through probability to an ad, if it is displayed in user's current session of her timeline. The training data are made up of all historical impressions shown across all users. These data are treated as i.i.d. and the learning algorithm optimizes for the global loss. An instance is represented as $(y, \mathbf{x})$, where $y \in \{\pm 1\}$ is the ground-truth binary label, with value 1 being the presence of clicks. Feature vector $\mathbf{x}$ is extracted from the ad, user, timeline, current session, and possible interactions between any two of the entities. These features should be general enough because no session repeats. Let $D = \{(y, \mathbf{x})\}$ be the set of all instances. The loss function for the pointwise learning can be formulated as [4]

$$L(\mathbf{w}, D) = \sum_{(y,\mathbf{x}) \in D} \ell\big(y, f(\mathbf{w}, \mathbf{x})\big)$$

where f is a hypothesis function, $\mathbf{w}$ is function parameters, and ℓ is a loss function for a single instance. In order to quickly capture user's change of information need, and enable large-scale online learning on the huge amount of click data, we use logistic regression to instantiate this learning framework with stochastic gradient descent (SGD) [4] as the optimization algorithm, a strategy also employed by Facebook [16]. Specifically, we have

$$\ell\big(y, f(\mathbf{w}, \mathbf{x})\big) = \log(1 + \exp(-yf(\mathbf{w}, \mathbf{x})))$$

where $f(\mathbf{w}, \mathbf{x}) = \mathbf{w}^T \mathbf{x}$, and $y \in \{\pm 1\}$.

4.2 Pairwise approach

In pointwise approaches, the learner takes as input a single instance one at a time, with presence of engagement as target value. Despite its advantage of directly minimizing prediction error, pointwise learning does not take into account the relative order of ads in terms of a particular user's preference, externalized by this user's click probability on each ad. However, as we know, ranking is critical in the auction of determining the winning ads – only top few candidates can be finally displayed, as we discussed as the second aspect of CTR prediction in Section 3.1.

Another significant advantage of directly learning user's relative ads preference is that we can address the training data sparsity challenge if we choose preference data wisely. It is natural that user's interest on ads can change over time. If a user clicked an ad a_A one year ago, and ignored an ad a_B today, it is doubtful to draw the conclusion that this user prefers a_A to a_B, due to a possible shift of interest of this user over a year. However, it is reasonable to assume that user preference is steady during a short time period. For example, two ads are more comparable if they are presented to the same user in one session. Two ads in the same session have almost the exact context, thus directly optimizing the perference order between them can address the sparsity challenge that ads are shown in different unique contexts.

In order to optimize for the relative user preference, we employ a pairwise learning approach [14], which attempts to incur less ranking loss. In particular, we train a pairwise model on ad pairs that are shown to one user in the same session. Let $P = \{((y_A, \mathbf{x}_A), (y_B, \mathbf{x}_B)) \,|y_A \neq y_B\}$ be the set of all pairs. The loss function is defined as

$$L(\mathbf{w}, P) = \sum_{((y_A,\mathbf{x}_A),(y_B,\mathbf{x}_B)) \in P} \ell\big(g(y_A - y_B), f(\mathbf{w}, \mathbf{x}_A) - f(\mathbf{w}, \mathbf{x}_B)\big)$$

where $g(y_A - y_B)$ transforms the difference of two individual instance labels into the label for pairwise learning. We use $g(y) = y/2$ to ensure that $g(y_A - y_B) \in \{\pm 1\}$. For logistic regression, $f(\mathbf{w}, \mathbf{x}_A) - f(\mathbf{w}, \mathbf{x}_B) = \mathbf{w}^T \mathbf{x}_A - \mathbf{w}^T \mathbf{x}_B = \mathbf{w}^T (\mathbf{x}_A - \mathbf{x}_B) = f(\mathbf{w}, \mathbf{x}_A - \mathbf{x}_B)$. Therefore, the logistic loss listed in classification section can still be used with no change [7], bringing the advantage that pairwise learning can be conducted in an online and scalable manner, just as pointwise learning.

4.2.1 Calibration

As stated above, pairwise approaches try to minimize ranking loss. Accordingly, the output of pairwise model is interpreted as preference score, rather than predicted click probability. However, estimation of click probability is indispensable for ad auctions, as described in Section 3.1. This means calibration is needed to transform score to click probability. A common practice is to use a sigmoid function [25], where the coefficients are learned through maximizing likelihood on the training set. An advantage of this transformation is that the relative order of instances ranked by score of the original model is preserved.

4.3 Combining pointwise and pairwise learning

Pointwise approaches try to obtain good estimate of click probability, while pairwise approaches aim to learn the ranking of impressions ordered by click probability. This brings their respective downside: pointwise methods are performing poorly on ranking, whereas pairwise methods tend to have the problem of inaccurate CTR estimation. Another practical problem could possibly arise if we resort to pairwise learning: not all sessions have more than one ad. This is especially true for Twitter, where the majority of auctions output no more than one winning ad. Consequently, a large proportion of instances would be wasted at training stage. The above analysis suggests that a combination of two approaches might be a good solution to simultaneously achieve good click probability estimation and ranking. Therefore, we propose an online algorithm, based on a combined optimization framework developed by Sculley [29]:

$$\min_{\mathbf{w}} \alpha L(\mathbf{w}, D) + (1-\alpha) L(\mathbf{w}, P) \tag{1}$$

with α being a trade-off parameter between optimizing towards classification and ranking. In [29], this trade-off is implemented by sampling an instance from D with probability α and a pair from P with probability $1-\alpha$. The sampling practice is perfect for offline static learning. For real online learning, the model receives training data in the form of ad stream, and ads of one session could return at different time points. Therefore, an algorithm adapted to the online setting has to be developed for the combined learning. We defer the description of the new algorithm to Section 5, where we introduce the infrastructure to support this learning approach.

One thing worth noting is that no calibration is necessary for combined learning, attributed to the classification component in the objective function.

4.3.1 Acquisition of more pairs

In practice, multiple ads shown in the same user session is still the minority case, because of the need to protect user experience by controlling the ads load. This is especially true for promoted Tweets, which are inserted into the main stream that users consume information from, unlike search ads or contextual ads displayed on the sidebar. As a result, only a small percentage of training instances fed to the model are from a pair of ads. Consequently, the learned model would be biased towards minimizing classification loss, failing to obtain enough pairs to induce a good ranker and mitigate the sparsity issue. To combat this problem, one strategy is to form more pairs artificially by grouping impressions from distinct requests. There are two grouping choices: across different users and within one user. Comparing impressions across users is to compare clicks collected from disparate preferences. It is possible to pair impressions from users sharing similar interests, borrowing the idea of collaborative filtering. For simplicity, in this work we only consider within-user grouping. Though a user's interest shifts over the course of time, it is reasonable to assume that each user's preference is stable within a short period of time. This makes it plausible to form "*pseudo-pairs*" by grouping impressions shown in different sessions but to the same user. To emphasize the time information, we attach importance weight to formed pairs based on time difference. Mathematically, let $S = \{((y_A, \mathbf{x}_A, t_A), (y_B, \mathbf{x}_B, t_B)) | y_A \neq y_B, t_A \neq t_B\}$ be the set of all pseudo-pairs, where t_A, t_B are the request time of impression A and B respectively. The loss function is defined as

$$L(\mathbf{w}, S) = \sum_{((y_A, \mathbf{x}_A, t_A), (y_B, \mathbf{x}_B, t_B)) \in S} \max\left(\min(\log \frac{N}{|t_A - t_B|}, 1), 0\right) \cdot \ell\big(g(y_A - y_B), f(\mathbf{w}, \mathbf{x}_A) - f(\mathbf{w}, \mathbf{x}_B)\big)$$

where N is acting as the size of a sliding window – the weight of a paired instance is 0 if $|t_A - t_B| \geq N$. The optimization framework incorporating pseudo-pairs can be formulated as:

$$\min_{\mathbf{w}} \alpha_1 L(\mathbf{w}, D) + \alpha_2 L(\mathbf{w}, P) + (1 - \alpha_1 - \alpha_2) L(\mathbf{w}, S) \tag{2}$$

4.4 Summary of methods

We give a summary of methods for future references.

1. **Pointwise learning**, which considers each ad as an individual training instance. This is the production model currently deployed at Twitter.
2. **Pairwise learning**. This method essentially refers to pairwise learning with calibration. Since the performance of pairwise approach before calibration is extremely poor in our empirical studies, we only report results on the calibrated version, which preserves the original ranking performance.
3. **Combined learning**, which takes into account classification and ranking under one framework.
4. **Pseudo**. Based on combined learning, we artificially create more pairwise training instances called pseudo-pairs.

5. ONLINE LEARNING INFRASTRUCTURE

In this section, we introduce the infrastructure at Twitter, an implementation used to conduct live experiments on the learning techniques described in Section 4. Considering the massive behavior data collected from each user's timeline, models that could be updated online are strongly necessary. In this section, we introduce how pointwise and combined learning could be conducted in an online manner, so that large-scale online A/B tests can be performed. Due to its poor estimation of CTR in offline study, we exclude pairwise learning in online experiments. Launching the method pseudo requires a significant change to Twitter's production system, which is unrealistic for this study. Therefore, pseudo is excluded as well. More detailed behavior analysis across all methods are performed via offline experiments.

5.1 Pointwise learning

Online learning requires we obtain new clicks and non-clicks in real-time so that new training instances could be formed to update the model. However, some difficulties surface due to the nature of stream.

The first issue has already been stated in Section 3.1, which is related to deciding whether users have seen the promoted Tweets. Since it is possible that users do not click ads simply because they fail to see it, only ads with impression callbacks will be considered as training examples.

There is yet another problem: the length of time varies for different users to finally see and engage with the promoted Tweets. This leads to a time difference for servers to receive engagement callbacks. The worst case is that users simply ignore these Tweets and servers can never obtain engagement callbacks. Hence comes

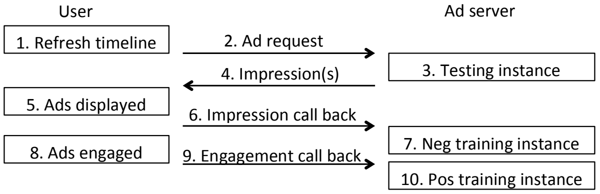


Figure 2: Online pointwise learning process

the problem of deciding the length of time the server should wait for user clicks. Because a training instance is not complete until its label is decided, this waiting time directly determines the lag of online learning. One solution to this problem is to cache impressions, and judge them as negative if no engagement callbacks are returned in a predefined amount of time, as did by Facebook [16]. That is to say, our judgment of labels could be wrong – engagement callbacks could return after the predefined time. The longer we cache ads, the more likely we obtain the ground-truth labels. However, trade-off exists – longer cache time leads to larger cache size and bigger delay of training. Our choice is not to cache at all – impressions are always set as negative and are added to the training set immediately when **impression** callback is received. If ever **engagement** callback returns, this impression is reset as positive to update the model again. This solution saves the large amount of cache space we have to offer and ensures no delay of training. Additionally, considering the rarity of clicking events, only a small percentage of examples need correction. Furthermore, preliminary work showed marginal improvement through caching. A summary of the online learning process is shown in Figure 2.

5.2 Combined learning

Combined learning requires training instances formed from both a single ad and a pair of ads. The procedure of obtaining single ads is identical to the one for pointwise learning. However two problems remain to be solved. First, we need to find a way to collect a pair of ads with click information. For simplicity, in the setting of online tests we only consider ads belonging to one session (request). Second, we have to decide how to trade-off between classification and ranking. That is, how to combine pointwise learning and pairwise learning.

The label for a pair can be decided only if we have click labels for both impressions in the pair. However, engagement callbacks return separately and the time of which varies a lot. To wait for the labels for both impressions, we have no choice this time but to cache them. The cache is accessed by using request id as the key. The cache value is a set of impressions with labels initialized to null. Each cache entry is guaranteed to be alive for 15 minutes. When an impression call back arrives, the label of the correspondent impression is set to negative. Whereas when an engagement call back returns, the label for the associated impression is turned to positive. Update of the model is only necessary when one impression's label changes from negative to positive, namely the moment when an engagement call back occurs. The positive instance is then paired with all negative instances belonging to one session. Chances are that some of the negative instances are clicked later, leading to input of false information to the model. In practice, we ignore such situations as we do not observe much performance loss, possibly due to the rarity of these circumstances.

With regard to the second problem, we assume a simple strategy – always apply pointwise learning for each individual impression, and do both pointwise and pairwise learning when there are more than one impression with differing labels. In this case, the trade-off parameter α in Equation 1 depends on the percentage of requests with a single ad and the number of clicked ads in requests with more than one ad.

Algorithm 1 Update $\mathbf{w}$ using combined learning

```
Input: $cache$, request ID $req\_id$, call back impression ID $imp\_id$, call back type $type$, current model parameter $\mathbf{w}$ , weight $w_p$ for paired instance
Output: Updated model parameter $\mathbf{w}$
$imp\_map \leftarrow cache.get(req\_id)$
$(y, \mathbf{x}) \leftarrow imp\_map.get(imp\_id)$ // get impression
if $type = impression\_call\_back$ then
    $imp\_map.set(imp\_id, (-1, \mathbf{x}))$ // set label to negative
    update $\mathbf{w}$ using $(-1, \mathbf{x})$ by SGD // pointwise learning
else // handle engagement call back
    $imp\_map.set(imp\_id, (+1, \mathbf{x}))$ // set label to positive
    update $\mathbf{w}$ using $(+1, \mathbf{x})$ by SGD // pointwise learning
    $P \leftarrow extract\_pairs(imp\_map, (+1, \mathbf{x}))$
    if $P.length > 0$ then // pairwise learning
        for Each pair $((y_A, \mathbf{x}_A), (y_B, \mathbf{x}_B))$ in $P$ do
            $\mathbf{x} \leftarrow (\mathbf{x}_A - \mathbf{x}_B)$
            $y \leftarrow g(y_A - y_B)$
            update $\mathbf{w}$ using $(y, \mathbf{x})$ and weight $w_p$ by SGD
        end for
    end if
end if
```

Therefore we would like to adjust α by changing the percentage of requests with more than one ad. Unfortunately decision of this percentage is complicated, and we have no direct control over it. However, we can still approximately change α by varying the weight w_p of instances formed by a pair of ads.

Note that paired instances are only used at training stage, and are excluded for prediction. In fact, for all approaches, prediction stage stays the same – we predict the click probability of each individual ad candidate at ad request time while running auctions.

Algorithm 1 gives a detailed description of the training procedure for combined learning. Algorithm 2 shows how pairs are extracted from a request. Line 3 to 8 guarantee that we generate positive and negative instances with equal probability.

Algorithm 2 $extract\_pairs$ (Extract pairs for a particular request)

```
Input: Impression map $imp\_map$, call back impression $(y, \mathbf{x})$
Output: An array of paired instances $P = \{((y_A, \mathbf{x}_A), (y_B, \mathbf{x}_B)) | y_A \neq y_B\}$
 1: $P \leftarrow \{\}$
 2: for Each negative instance $(y^-, \mathbf{x}^-)$ in $imp\_map$ do
 3:     Draw $z$ uniformly at random from $[0, 1)$
 4:     if $z < 0.5$ then
 5:         Form a pair $p \leftarrow ((y, \mathbf{x}), (y^-, \mathbf{x}^-))$
 6:     else
 7:         Form a pair $p \leftarrow ((y^-, \mathbf{x}^-), (y, \mathbf{x}))$
 8:     end if
 9:     $P \leftarrow P \cup \{p\}$
10: end for
```

6. OFFLINE EXPERIMENT

We evaluate the methods using both offline and online experiments. Online experiments are used to verify the true effectiveness of methods in real setting, while more detailed analysis is done through offline simulation. We introduce the offline experiments in this section, and describe the online tests in Section 7.

6.1 Metrics

6.1.1 NRIG

In order to quantify the accuracy of predicted click probability, we use normalized relative information gain (NRIG), based on relative information gain (RIG) [15]. Given test set T, the empirical CTR of the data is defined as $\bar{p} = \sum_{t=1}^{T} y_t/|T|$, where $y_t \in \{0, 1\}$ is the ground-truth label. The entropy is referred as $H(\bar{p}) = -(\bar{p} \log \bar{p} + (1 - \bar{p}) \log(1 - \bar{p}))$. The empirical cross entropy for a model is $CE = -\frac{1}{|T|} \sum_{t=1}^{T} (y_t \log p_t + (1 - y_t) \log(1 - p_t))$, where p_t is the model predicted click probability for t-th impression. Then relative information gain could be computed as $RIG = (H(\bar{p}) - CE)/H(\bar{p})$. To normalize RIG, we compute normalized prediction $\hat{p}_t = p_t \times \dfrac{\bar{p}}{\sum_{t=1}^{T} p_t/|T|}$ This ensures that the average of the normalized prediction equals to $\bar{p}$, the empirical CTR. The subsequent computation of NRIG is almost identical to RIG, with the exception that p_t is replaced by $\hat{p}_t$ when computing CE.

6.1.2 AUC

To measure ranking quality, we compute the area under receiver operator curve (AUC) [13] for the subset of requests containing more than one ad. For each of such requests, we calculate AUC using (1) predicted ranking, formed by the estimated click probability output by the model, and (2) ground-truth, which is the presence or absence of engagements for each ad in the request. We report final AUC as an average over requests with more than one ad.

6.2 Procedure

Instead of k-fold cross-validation, we use an evaluation procedure based on progressive validation [5]. This procedure works in a streamed fashion, simulating the online setting described in Section 5: both test and training data come in a stream, and instances are first used for testing at request time, and are later used for training.

With regard to pseudo, a method not covered in Section 5, we do offline simulation using the following process. Cache time is set to N, the sliding window size. Impressions displayed to the same user that coincide in the same window can form pairs. Similar to combined learning, a positive ad is paired with each of all negative ones.

6.3 Dataset

To conduct offline experiments, we collected data from a random week of year 2014. Given that progressive validation is employed, there is no need to partition the data into training and testing. To give readers a sense of the scale of the dataset, in the third quarter of 2014, there are 181 billion timeline views on Twitter,[2] approximating the number of ad requests.

6.4 Parameters

All parameters, including learning rate of stochastic gradient, parameter controlling regularization, coefficients of calibration model, and the weight w_p of instances formed by a pair of ads, are tuned using the first day's data. This period is excluded in the reports of experiment results. In addition to parameter tuning, the first day training allows the models to warm up.

In combined learning, we have a parameter α to control the importance of classification error against ranking error. As stated in Section 5, we adjust it by varying the weight w_p of instances formed by a pair of ads. In pseudo, we have parameter N, the size of sliding window to form pseudo pairs. Both w_p and N are tuned to optimum using the data of first day, equaling to 20 and 8 hours respectively. Actually in pseudo, we have another two parameters α_1 and α_2 in Equation 2, controlling the importance of single ads, paired ads, and pseudo-pairs. Like α, the importance of pseudo-pairs could be approximated by the maximum possible weight of pseudo-pairs. Due to its similarity to α, we omit the study of it by simply setting the maximum possible weight to 1. A sensitivity study of parameter α and w_p will be performed later in the experiment result section.

6.5 Features

In order to better catch user's information need, deal with training sparsity problem and capture the unique properties of ads in streams, we design features from four perspectives: ad, user, ad-user interaction, and context of the stream. Unfortunately, we can only give a brief description of features in each category, without disclosing the specific features to be used.

- Ad. Features related to an ad can be further divided into three subcategories. 1) Advertiser profile, such as a set of topics pertinent to an advertiser. 2) Meta information, including ID of an advertiser, type of ad, etc. 3) Click history, an example is past clicks of an ad on the population level.

- User. Analogous to ad, user related features also fall into three subcategories. 1) User profile, such as user related topics. 2) Meta information, including gender and device information. 3) Click history, such as advertisers engaged by a user in the past.

- Ad-user interaction. The interactions can be characterized by ad-user similarity and user click history of ads belonging to the same advertiser. Multiple similarity measures are included, e.g., similarity between user and advertiser's profile, and whether the user is related to certain keywords specified by the advertiser.

- Context of the stream. The context refers to features available within a session, which enables us to find out interactions between ads and organic messages. Because no sessions repeat, these contextual features have to be general enough. Specifically, we include positions of the promoted Tweets, used to correct position bias at training time, and similarity between the promoted Tweets and organic Tweets in one session. To compute the similarity, we consider two forms of text representations: bag of words and word vectors calculated by word2vec [22].

Continuous features are transformed to categorical ones using boosted decision trees, a similar discretization technique reported by Facebook [16].

6.6 Experiment results

6.6.1 Overall performance

Table 1 summarizes performance of different approaches, averaged over one week but excluding the first day. The result is reported based on percentage of improvement using pointwise learning, our production model, as the reference. Though we are not aware of any statistic tests designed for NRIG, empirical studies show that the improvement listed in Table 1 is strongly significant.

Consistent with our expectation, pairwise learning scores relatively high in terms of AUC, but does poorly on NRIG, even with calibration. Therefore, pairwise learning indeed gives a good ranking of ads, but fail at estimating the click probability. This is due to its objective of incurring less ranking loss by attempting to give a correct order of ads, however without taking into any account of the accurate estimation of click probability.

[2] https://investor.twitterinc.com/releasedetail.cfm?releaseid=878170

Method	Pairwise	Combined	Pseudo
Relative NRIG (%)	-75.90	$+9.44$	$+10.25$
Relative AUC (%)	$+1.76^{***}$	$+1.91^{***}$	$+2.11^{***}$

Table 1: Summary of performance relative to pointwise learning method. * indicates the improvement over Pointwise method is statistically significant according to Wilcoxon signed rank test at the significance level of 0.001**

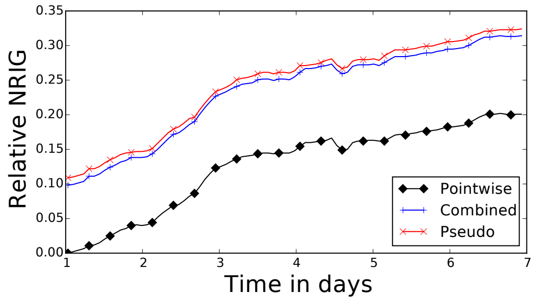


Figure 3: Relative NRIG of one week excluding the first day. The pairwise method is excluded due to its extremely low score.

In contrast, the pointwise model achieves reasonable result on NRIG, but is inferior to all the other methods on ranking. This behavior can be explained by its optimization purely towards minimizing errors of CTR prediction, while ignoring relative ranking of ads.

Combining pointwise and pairwise methods brings a huge performance gain on both CTR estimation and ranking. We suspect the reason behind it is that learning two tasks simultaneously requires the model to put more weight on features that could achieve good CTR prediction as well as good ranking. The enhancement of both NRIG and AUC indicate that combining pointwise and pairwise learning-to-rank techniques indeed helps us better capture user's current interest, and alleviates the training sparsity problem by incorporating relative preference information.

The last column of Table 1 shows the results after adding pseudo-pairs to the combined model. We could find a further improvement over the original combined model. This indicates that these pseudo-pairs could contribute positively, which is good news if we are concerned that showing too many ads in one request could hurt user experience. Pseudo-pairs alleviate this problem and corroborate the assumption that user preference are stable in a short period of time, and can be utilized to capture user's short-term interest.

6.6.2 *Learning behavior*

In addition to the average performance, we are also interested in how different methods behave over the course of time. Figures 3 and 4 show such behavior changes in terms of relative NRIG and AUC. The relative value is computed as the improvement over the performance of pointwise learning at the end of the first day. The pairwise method is excluded in Figure 3, due to its extremely low score and resemblance to other methods in terms of the learning curve.

For both metrics, all the methods exhibit similar patterns: an evident rise of performance in the first day, followed by small, gradual but continual improvement. Though not completely stabilized at last, the increase shows a tendency to level off. This common pattern depicts the learning behaviors of the online models: the effectiveness of the model keeps improving as more and more training examples are revealed. The improvement starts to diminish as the model saturates.

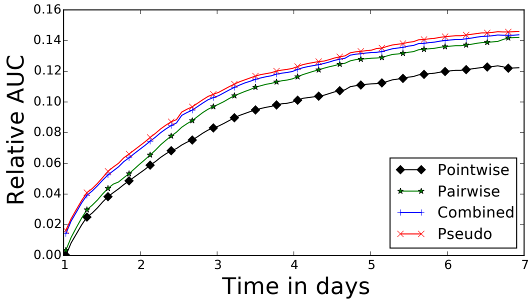
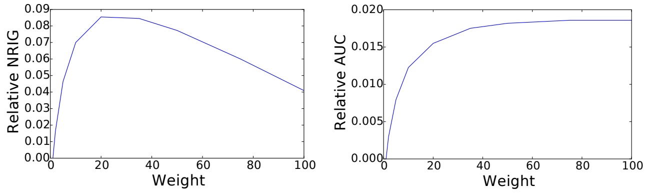


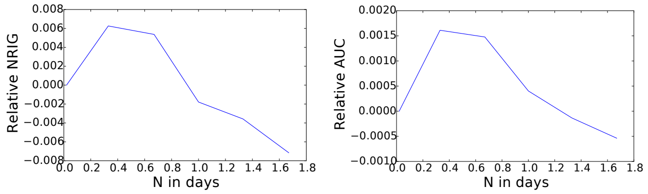
Figure 4: Relative AUC of one week excluding the first day.



(a) NRIG

(b) AUC

Figure 5: Change of performance against weight.



(a) NRIG

(b) AUC

Figure 6: Change of performance against N.

Consistent with our findings from the average performance, pseudo is the winner on both CTR prediction and ranking – it persistently performs better than all other methods across the entire learning process. Pairwise learning achieves reasonable results on AUC, but is way behind other methods on NRIG.

6.6.3 *Parameter sensitivity*

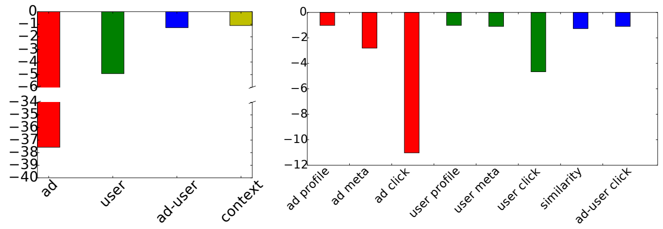
We study two parameters offline: the weight w_p of instances formed by a pair of ads and sliding window size N to form pseudo pairs. The performance changes are plotted in Figures 5 and 6.

Intuitively, the higher the value of w_p, the more likely the model would be optimized towards ranking. This intuition is supported by the experiment result: we observe improved AUC as paired instance weight w_p increases. This improvement flattens out finally. In contrast, The change of NRIG resembles a right-skewed bell shape – an initial increase to a certain point, followed by a decline when weight w_p gets too large. The increase at the beginning might be due to that information brought in by ranking is beneficial to estimation of CTR as well. However, as the model keeps shifting its focus on optimizing towards ranking, the classification quality is finally undermined.

When we vary the size of sliding window N, the performance of classification and ranking exhibit similar patterns. At the beginning, there is an increase of performance resulting from the inclusion of more pairs. However, as the size of sliding window becomes even larger, the model degenerates, likely caused by the formation of training instances from two ads with a bigger time difference. User preference might have changed as time passes by, making two ads incomparable. It might be surprising that the best N is around 8 hours. Actually a user's intent could change rapidly in daily tasks (e.g., choosing restaurants). The context of timeline ads also changes fast and affects the user's perception of ads.

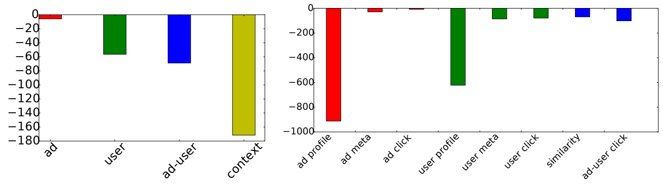
6.7 Feature analysis

As mentioned before, we put features into four categories: advertiser, user, ad-user interaction, and context of the session. To study the importance of different feature categories, we rerun the offline experiments of combined learning by removing one feature category at one time, while keeping all other features present. Figures 7 and 9 show the negated percentage of performance reduction relative to the method using the entire feature set. We negate the values in order to make the figures more graphically intuitive. Each category is colored distinctively, and features belonging to the same major category carry the same color. The left figure plots the result after removing all features belonging to a major category, whereas the right figure is only removing a subcategory. Apart from removing features, we also consider using a single category alone, with all other features excluded. The results are shown in Figure 8 and 10.



(a) Major categories (b) Subcategories

Figure 7: Negated percentage of NRIG diminished when removing features of a particular category.



(a) Major categories (b) Subcategories

Figure 8: Negated percentage of NRIG diminished when keeping features of a particular category.

We could analyze feature importance by reading the two types of figures side by side. There are in total four possible outcomes, which are summarized in Table 2.

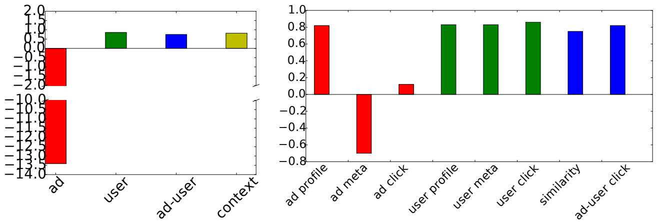
6.7.1 NRIG

Looking at the major categories depicted in Figure 7 (a) and 8 (a), we can see that in order to achieve a good CTR prediction performance, the ad features play an importance role. Removing user features leads to a significant decrease of model quality as well, but is much less severe than that of ad features. Excluding ad-user features causes a counterintuitively marginal impact. By looking at Figure 8 (a), where ad-user features achieve comparable performance as user features, we could conjecture that features from ad-user category correlate with ad and user features – if a user clicks frequently and an ad is popular, there is usually a good interaction between the user and the ad. Context information extracted from a session is not so useful, suggesting that textual information in the context is hard to capture user's current interest.

Remove	Keep	Interpretation
L	S	Important features
L	L	Does not happen
S	L	Weak features
S	S	Important but possibly covered by other features

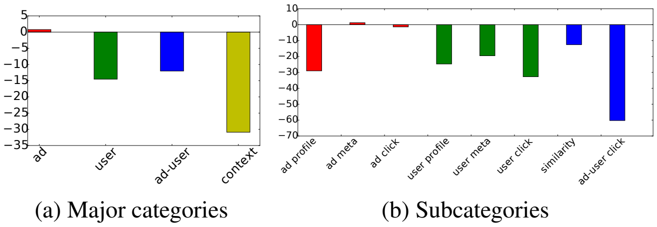
Table 2: Possible outcomes for each feature category. "L" under "Remove" column means there is a large drop of performance after removing a feature category. Likewise, "S" means small.

Investigating the subcategories by reading Figure 7 (b) and 8 (b), click history of both advertisers and users are most important, confirming the result reported by Facebook [16]. Somehow surprisingly, ad meta information imposes a non-negligible effect on CTR estimation. This is due to the existence of advertiser IDs, which act as an indirect way of counting the popularity of advertisers.



(a) Major categories (b) Subcategories

Figure 9: Negated percentage of AUC diminished when removing features of a particular category.



(a) Major categories (b) Subcategories

Figure 10: Negated percentage of AUC diminished when keeping features of a particular category.

6.7.2 AUC

When we study feature importance on ranking, one unexpected observation in Figure 9 and 10 is the existence of positive values, which means AUC is actually improved after removing a feature category or when applying a single category alone. We make two comments on this observation. First, the majority of instances fed to the learning model are formed from single ads, rather than a pair of ads. In other words, the model would optimize towards CTR estimation more than ranking. Consequently, learned features might contribute positively to CTR prediction, even if ranking performance is sacrificed. Second, the sacrifice of AUC is not significantly large – all positive values are no more than 1%.

Having discussed about the abnormality, we move our focus to the feature importance with respect to ranking. We can see from

Method	Pointwise	Combined
Relative CTR (%)	+14.59	+26.10(+10.05)
Relative RPMq (%)	+57.20	+57.89(+0.44)

Table 3: Summary of performance relative to random-pointwise. Numbers in the parentheses indicate performance of combined learning relative to pointwise learning.

Figure 9 (a) and 10 (a) that the ad features are paramount in deciding rankings. However, except for the ad category, removing any other single category causes almost no effect on ranking. Judged from Figure 10 (a), user information and ad-user interaction can to some extent be helpful in ranking. But removing either of them is not hurting the performance, due to the power of ad features. It is reasonable that context features are not strong indicators, as ads in the same session almost face the same context.

According to Figure 9 (b) and 10 (b), excluding a single subcategory does no harm to ranking, as either ad meta or ad click information is strong enough to maintain a good performance. Aside from ad features, similarity between an advertiser and a user could give a relatively good ranking while clicks of a user towards a particular advertiser cannot. The reason is that we have much richer information to calculate similarity, including user's following accounts, and user's Tweets. However, click information is much sparser – typically users seldom click ads, not to mention ads from a particular advertiser.

7. ONLINE EXPERIMENT

7.1 Metrics

For online experiment, we compare the click-through rate (CTR) and revenue per thousand requests (RPMq) of different approaches. We report both metrics because there is a trade-off between CTR and RPMq. A model can play conservatively by only placing ads for requests that are most likely to result in a click, while ignoring most ad requests. In this case, this model could achieve a rather high CTR, but at the cost of RPMq. Conversely, if a model acts aggressively by sending out ads to every request, it could possibly increase RPMq by sacrificing CTR, which hurts user experience and causes negative effects for the company in the long run.

Consequently, a model can be safely claimed to be successful if it can improve one of the metrics, while keeping the other one at least as good as competing methods.

7.2 Procedure

A random week of year 2014 is selected to evaluate pointwise learning and combined learning via online A/B tests. Additionally, a weaker baseline **random-pointwise** is introduced, which adds some randomness in the ad selection process to the pointwise learning model. Users are randomized into different bins by hashing user IDs, with one bin corresponds to 0.5% of all Twitter users. Each method is then evaluated on users belonging to one of the bins.

7.3 Experiment results

To further evaluate the effectiveness of combined learning in real setting, we launched the online experiments, comparing it with the baseline pointwise learning and a weaker baseline random-pointwise. The result averaged over one week can be referred in Table 3.

The performance on each individual day are plotted in Figures 11 and 12, using the result of random-pointwise on the first day as reference. A few observations could be made from the figures. First, both pointwise and combined learning are acting better than random-pointwise, suggesting the benefits of taking learning approaches towards advertising in stream. Second, combined learning strictly dominates and significantly outperforms pointwise learning on CTR. In terms of RPMq, it performs on par with, and on average slightly better than pointwise learning. As described in Section 7.1, trade-off exists between CTR and RPMq. Improving CTR while maintaining RPMq has proved the superiority of combined learning over pointwise learning. More specifically, the results mean that combined model is showing fewer ads to users. But these ads lead to higher click-throughs, compensating for the revenue loss caused by placing fewer number of ads. Though short-term revenue brought by the two methods stays close, user experience is improved when using combined learning, bringing in more long-term benefits to the company. Moreover, some threshold parameters could be adjusted in the auction system, so that the combined model could distribute more ads. In this way, combined learning can achieve similar CTR score as the pointwise baseline, but with much higher revenue, measured by RPMq.

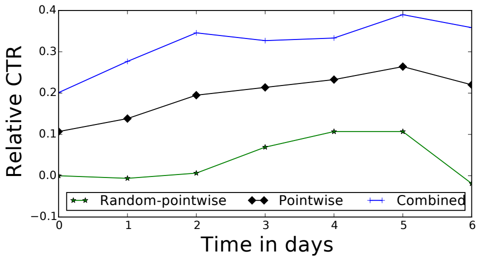


Figure 11: Relative CTR as a function of time.

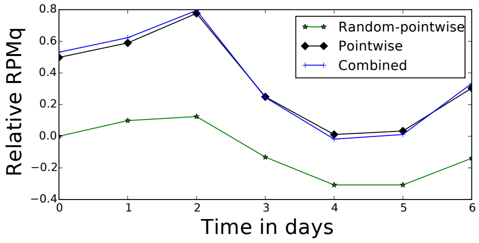


Figure 12: Relative RPMq as a function of time.

8. CONCLUSION

In this paper, we presented the problem of click-through prediction for advertising in Twitter Timeline. Compared with traditional computational advertising *sponsored search* and *contextual advertising*, placing ads in a Tweet stream is particularly challenging given the nature of a data stream: the context in which an ad can be placed changes dynamically and few ads could be placed in the same session. This makes the information available for training extremely sparse.

We proposed a learning-to-rank method which not only addresses the sparsity of training signals but also can be trained in real-time with great scalability. The proposed method was evaluated using both offline experiments and online A/B tests, involving very large collections of Twitter data and real Twitter users. We demonstrated that the proposed method not only improved prediction performance in offline simulations but also significantly enhanced actual CTR when deployed to the real Twitter ads-serving system, using the production model as the baseline.

9. REFERENCES

[1] R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, H. Jiang, T. Qiu, A. Reznichenko, D. Ryabkov, M. Singh, and S. Venkataraman. Photon: Fault-tolerant and scalable joining of continuous data streams. In *Proceedings of the 2013 international conference on Management of data*, pages 577–588. ACM, 2013.

[2] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Information retrieval*, 13(3):291–314, 2010.

[3] J. Battelle. *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*. Penguin, 2005.

[4] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[5] A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208. ACM, 1999.

[6] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 559–566. ACM, 2007.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[8] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *Proceedings of the 17th international conference on World Wide Web*, pages 417–426. ACM, 2008.

[9] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *Proceedings of the 17th international conference on World Wide Web*, pages 227–236. ACM, 2008.

[10] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 295–303. Association for Computational Linguistics, 2010.

[11] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. Technical report, National Bureau of Economic Research, 2005.

[12] D. S. Evans. The online advertising industry: Economics, evolution, and privacy. *The journal of economic perspectives*, pages 37–60, 2009.

[13] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[14] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[15] T. Graepel, J. Q. Candela, T. Borchert, and R. 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.

[16] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1–9. ACM, 2014.

[17] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison. Learning to rank social update streams. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 651–660. ACM, 2012.

[18] B. J. Jansen and T. Mullen. Sponsored search: an overview of the concept, history, and technology. *International Journal of Electronic Business*, 6(2):114–131, 2008.

[19] M. Karimzadehgan, W. Li, R. Zhang, and J. Mao. A stochastic learning-to-rank algorithm and its application to contextual advertising. In *Proceedings of the 20th international conference on World wide web*, pages 377–386. ACM, 2011.

[20] A. Lacerda, M. Cristo, M. A. Gonçalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 549–556. ACM, 2006.

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

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[23] V. Murdock, M. Ciaramita, and V. Plachouras. A noisy-channel approach to contextual advertising. In *Proceedings of the 1st international workshop on Data mining and audience intelligence for advertising*, pages 21–27. ACM, 2007.

[24] T.-T. Nguyen, T.-T. Nguyen, and Q.-T. Ha. Applying hidden topics in ranking social update streams on twitter. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on*, pages 180–185. IEEE, 2013.

[25] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. Citeseer, 1999.

[26] A. Ratnaparkhi. A hidden class page-ad probability model for contextual advertising. In *WWW Workshop*, 2008.

[27] B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. Silva de Moura. Impedance coupling in content-targeted advertising. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 496–503. ACM, 2005.

[28] M. Richardson, E. Dominowska, and R. 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.

[29] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 979–988. ACM, 2010.

[30] J. G. Shanahan and G. Kurra. Digital advertising: An information scientist's perspective. In *Advanced Topics in Information Retrieval*, pages 209–237. Springer, 2011.

[31] X. Zhang, B. He, and T. Luo. Transductive learning for real-time twitter search. In *ICWSM*, 2012.