Serving Ads to “Yahoo Answers” Occasional Visitors

Michal Aharon
Yahoo Labs, Haifa, Israel
michala@yahoo-inc.com

Amit Kagian
Yahoo Labs, Haifa, Israel
akagian@yahoo-inc.com

Yohay Kaplan[*]
Technion, Haifa, Israel
yohayk@cs.technion.ac.il

Raz Nissim
Yahoo Labs, Haifa, Israel
raz@yahoo-inc.com

Oren Somekh
Yahoo Labs, Haifa, Israel
orens@yahoo-inc.com

ABSTRACT

Modern ad serving systems can benefit when allowed to accumulate user information and use it as part of the serving algorithm. However, this often does not coincide with how the web is used. Many domains will see users for only brief interactions, as users enter a domain through a search result or social media link and then leave. Having access to little or no user information and no ability to assemble a user profile over a prolonged period of use, we would still like to leverage the information we have to the best of our ability.

In this paper we attempt several methods of improving ad serving for occasional users, including leveraging user information that is still available, content analysis of the page, information about the page's content generators and historical breakdown of visits to the page. We compare and combine these methods in a framework of a collaborative filtering algorithm, test them on real data collected from Yahoo Answers, and achieve significant improvements over baseline algorithms.

1. INTRODUCTION

Online advertising is booming in recent years and is expected to keep on growing dramatically in the future. Modern web scale ad serving platforms manage a vibrant electronic marketplace where advertisers introduce their campaigns and bids, and ads are matched dynamically to users according to their inferred interests and previous interactions [8]. Such marketplaces rely heavily on the ability of the system to accurately and quickly predict click probability. Machine learning techniques in general, and especially personalization and recommendation algorithms, are proven to be quite successful at the task of predicting ad click probability.

One of the most successful approaches for personalization and recommendation tasks is *collaborative filtering* (CF) [18]. This technique relies only on past user behavior (e.g., previous transactions or feedback) and does not require the creation of explicit profiles. It requires no domain knowledge or content analysis, and excels at exploiting popularity trends, which drive much of the observed users' interactions. A fundamental problem that arises when employing CF techniques is the cold-start problem. In a nutshell, the cold-start problem is caused by the system's incapability of dealing with new items or new users due to the lack of relevant transaction history.

In [2], a novel approach for ad ranking named OFF-Set was introduced. This CF low-ranked matrix factorization (MF) based algorithm is designed to handle an extreme setting of perpetual cold-start problem where users are being encountered only a few times by the system. OFF-Set deals with this extreme scenario by representing users as a combination of latent vectors of their demographic features (such as age, gender, geo, and device).

Here, we consider an even more severe scenario than that of [2], where basic user demographic features may be unknown as well. This scenario is quite common at the popular Yahoo Answers website, where a significant portion of traffic is generated by external search engines. In these cases, ad personalization (or even segmentation) via ad click prediction becomes a challenging task due to insufficient user information.

To mitigate the lack of user information and limited user feedback, we take advantage of the fact that the user has chosen to visit this specific Yahoo Answers page that includes a collection of top rated answers to a specific question (see Section 3). It may be that this page was ranked high in a search results page or that a Yahoo Answers user wanted to answer a specific question in his area of interest or expertise. Therefore, we can infer some of the missing demographic information by incorporating contextual attributes and features extracted from the content of the visited Yahoo Answers page. These features are then used by a standard CF low-ranked MF latent model to predict the ad click probability. Offline experimental evaluation is performed to demonstrate the benefits of our algorithms over several simple baselines. For the evaluation process we have used data collected from real Yahoo Answers pages and actual Yahoo Answers traffic. The evaluation is done incrementally so the contribution of individual features can be assessed.

The rest of this work is organized as follows. Section 2 briefly surveys related work. Background on Yahoo Answers and LDA is provided in Section 3. A formal definition of the ad serving problem at hand is included in Section 4. Section 5 holds a detailed explanation of our selected approach for modeling, while baseline algorithms are described in Section 6. The experimental setup, dataset, and results of our evaluation are presented in Section 7. We conclude in Section 8.

[*]This work was done during a summer internship at Yahoo Labs.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2741997.

2. RELATED WORK

Collaborative Filtering.

As mentioned earlier, CF is one of the most successful techniques for constructing a recommender system. The two major fields of CF are *neighborhood methods* and *Latent Factor Models* (LFM). Neighborhood methods compute the relationships among users or between items. For example, the item-item approach [20, 26] predicts the user's preference of an item by inspecting her ratings of "neighboring" items (items that are rated similarly by users).

LFM characterizes both items and users as vectors in a space which is inferred from observed data patterns. The latent space representations strive to capture the semantics of items and users, which are derived from their observed interactions. One of the most successful realizations of LFM, which combines good scalability with predictive accuracy, is based on low-rank MF (e.g., see [18]). In particular, low-rank MF provides substantial expressive power that allows modeling specific data characteristics such as temporal effects [17], item taxonomy [7], and attributes [1]. Recently, an online low-rank MF based CF algorithm was also used for ad ranking [2]. In this paper we use an approach similar to that of [2] and predict ad click probabilities of Yahoo Answers occasional users.

Cold-Start Problem of CF techniques.

An inherent limitation of CF is the need for historical user-item interactions. When such history is limited, CF-based recommenders cannot reliably model new entities, leading to the item and user cold-start problems. Despite users and items being usually represented similarly in the latent space, these two problems are essentially different. The main difference comes from the ability to interview new users when joining the service in order to bootstrap their modeling. Another difference is that in most settings the number of users is much larger than the number of items. Hence, a typical item usually gets more ratings than an individual user provides.

To mitigate the item cold-start problem of CF, a common practice involves utilizing external content on top of users' feedback. The basic approach is to leverage item attributes and combine them with the CF model in a way that allows treating new items [1, 12, 13, 22]. In [1] the authors proposed a regression-based latent factor model for cold-start recommendation. In a later work [22], the authors improve on [1] by solving a convex optimization problem that estimates the weight matrices, instead of computing the low-rank matrix decomposition of [1]. Another approach to tackle the item cold-start issue by combining content information with CF, based on *Boltzmann machines*, is considered in [12, 13]. When no item content or context information is available new items' latent vectors can be estimated by a linear combination of the latent vectors of their raters and their respective ratings [23, 16, 3, 4].

Modeling the preferences of new users can be done most effectively by asking them to rate several carefully selected items of a seed set during a short interview [15, 24, 25, 11]. Item seed sets were constructed according to various criteria such as *popularity* (items should be known to the users), *contention* (items should be indicative of users' tendencies), and *coverage* (items should possess predictive power on other items). The importance of adaptive interviewing process using adaptive seed sets was already recognized in [24]. Adaptive interviewing is commonly implemented by decision trees algorithms [25, 19, 10]. A method for solving the problem of initial interview construction within the context of learning user and item profiles is presented in [28].

Ad Click Prediction.

As mentioned earlier, machine learning techniques in general, and especially personalization and recommendation algorithms, are proven to be quite successful for the task of personalized ad ranking. A selection of case studies and topics drawn from recent experiments in the setting of a deployed ad click prediction system by Google are presented in [21]. These include improvements in the context of traditional supervised learning based on a *Follow the regularized Leader* (FTRL)-Proximal online learning algorithm (which has excellent sparsity and convergence properties) and the use of per-coordinate learning rates. Practical lessons from experimenting with Facebook ads data are presented in [14]. Explore/ exploit approach using multi-armed bandits algorithms (i.e., *Thompson Sampling*) combined with standard regularized regression model is used in [6] to predict ad click probability. Recently, an online low-rank MF based CF algorithm was also used to compute personalized ad ranking [2]. To overcome data sparsity (ad click probability are of an order of magnitude smaller than news click probability) ads are represented by latent vectors while users latent vectors are constructed from their features' (such as age and gender) latent vectors.

3. BACKGROUND

Yahoo Answers.

Yahoo Answers is a platform for users (askers) to post questions and to get answers from other users (answerers). It has millions of active users who interact on a large variety of topics. A user can post any question in free language and the provided answers are typically ranked by users in order to have the best answers easily available. Users also assign their questions to a specific category within a predefined category hierarchy. For example, the question *how to stop my dog from barking?* was assigned to the category *Pets/Dogs*.

A substantial portion of Yahoo Answer's traffic is composed of users that are redirected from commercial search engines. A typical user will first submit a question into a search engine and later click a search result that leads to an already answered question in Yahoo Answers. As a result of this behavior, a considerable portion of the users that visit Yahoo Answers pages are occasional users that are unfamiliar to the page. This creates a user cold-start challenge for ad-serving: regularly having a vast amount of users with very limited information known about them. We address this challenge by relying on user features from click modeling as described below.

Latent Dirichlet Allocation (LDA).

LDA is a generative probabilistic model of a corpus, originally proposed for modeling text documents [5]. The intuition behind this model is that documents exhibit multiple topics, which are distributions over a fixed vocabulary W. The LDA algorithm performs data analysis to compute an approximation of the conditional distributions of the hidden variables (*topics*), given the observed variables (words of the documents). In its standard application, LDA gets as input a set of documents, along with their bag-of-words representation, often represented as a sparse matrix in which the (i, d) entry is the number of times the word i appeared in the document d. It produces a set of n topics, where each topic is a multinomial distribution vector in $R^{|W|}$ representing the probability of each word to appear in a document discussing that topic. The documents' latent topics can later be inferred according to the words they contain. Here, we apply LDA to Yahoo Answers pages' content (see [27]).

4. PROBLEM DEFINITION

Each 'event' in our data is represented by a tuple (u, a, I_C), that consists of a user u, an ad a and an action indicator $I_C \in \{0, 1\}$ (either a click or a skip). An event for which $I_C = 1$ is called a 'click event'. We refer to $\mathcal{C}$ as the set of all click events, and $\mathcal{I}$ as the set of all events.

The problem setting is as follows: The algorithm gets as input an ordered set of events (the ordering is done by the time of the event). For each such event, the algorithm first provides a prediction for the CTR probability, and only afterwards uses this event to update the model and tune it for future predictions. The provided probability is later used to evaluate the algorithm's performance. This settings simulates an on-line learning approach in which the algorithm performs a single pass over the events.

The goal of the algorithm is to maximize the *Area Under a ROC Curve* (AUC) value (that reflects the ad ranking quality). For binary scores, the AUC value is equivalent to the probability that a randomly chosen positive event (click) is ranked higher than a randomly chosen negative event (non-click impression). The AUC value ranges between 0-1, where a random model has an AUC score of 0.5. Further details, including an algorithm to compute AUC, is provided in [9].

5. MODELING

In this section we present our approach for Click-Through Rate (CTR) prediction using collaborative filtering. We later describe how we applied this approach on Yahoo Answers data.

5.1 Collaborative Filtering for CTR Modeling

5.1.1 Predicted Click-Through Rate

In many online advertising platforms, most served ads use the *Cost-Per-Click* (CPC) pricing type mechanism, i.e., an advertiser pays only when its ads are clicked. For simplicity reasons we will refer to this pricing type mechanism only. In a CPC pricing system, the revenue received from each click on an ad is approximated by the bid that the advertiser sets for this ad. The approximation is due to the *Generalized Second Price* mechanism used by many advertising platforms [8], as well as other considerations that are out of the scope of this paper.

$$revenue \approx \sum_{(user,ad) \in C} bid(ad) = \sum_{(user,ad) \in I} CTR(user,ad) \cdot bid(ad)$$

where C is the set of click events, and I is the set of all events and $CTR(user, ad)$ is the click-through rate of a given user on a given ad. Since the bids are known in advance, we would like to compute a model which predicts the click probability of user u on ad a, denoted by $pCTR(u, a)$ (predicted CTR).

5.1.2 Latent Factor Modeling

In order to generate a model for CTR prediction, we apply an online collaborative filtering approach. We train a *latent factor model* in which each user and each ad are represented by a vector of dimension *d*. If $p_u \in R^d$ is the vector representing a user u, and $p_a \in R^d$ is the vector representing an ad a, then the probability of u to click on ad a is approximated by applying a sigmoid function over the inner product of the two corresponding vectors,

$$pCTR(u, a) = \frac{1}{1 + exp^{-(b + \langle p_u, p_a \rangle)}}$$

where b is the model's bias. Note that this sigmoid function is monotonous increasing and outputs values in the range [0,1].

The model is trained to maximize the log-likelihood of the training data, using Stochastic Gradient Descent (SGD), to optimize the following LogLoss function,

$$\sum_{\{u,a,I_C\} \in I} -I_C \cdot \log pCTR(u,a) - (1 - I_C) \cdot (1 - \log pCTR(u,a))$$

5.1.3 Feature Based User Representation

There are hundreds of millions of users in our dataset, however, most of the users do not click on ads at all. Therefore, we refer to user features in order to construct latent user representations even for users that have never clicked on any ad. For example, we may employ the fact that a user u is a 20 year old female from New York in order to predict u's click probability on a given ad.

Each user feature is represented in the model with its own latent vector $p_f \in R^d$. The vectors corresponding to a user's set of features are averaged to construct the final vector

$$p_u = \frac{1}{k} \sum_{i=1}^{k} p_{f_i}$$

where $\{f_1, ..., f_k\}$ is the set of k features describing user u. The latent vectors of the features are trained using SGD with the above LogLoss function.

5.2 Click Modeling for Yahoo Answers

Our approach was to combine several methods of generating or approximating user information. The initial set of user features was all the available user information. For each event we had access to the following user information:

User location - Domain size of a few thousands with 100% coverage, derived from the IP address used by the user.

User device - The type of device the user is using. Domain size is 150 with 100% coverage.

Age - User age, bucketed into a domain of size 12 with 25% coverage.

To compliment the available user information, we characterize the user by the page that he visits, therefore further characterizing the user as 'a user that is interested in this page'. Using this approach, the user features can be expanded using available features of the pages of Yahoo answers. The following features were considered as additional user features under the CF framework:

Category - The category the page is associated with on Yahoo Answers. There are about~2000 categories arranged in a hierarchy with 30 categories at the top level. The algorithm used these top level categories as a feature.

Page content LDA - An LDA topic breakdown of the content of the page. There are ~3000 possible LDA topics. Each page is associated with up to 15 top topics along with a weight measure for each (sum of these weights < 1).

To handle features such as LDA topics, in which there are several weighted entries per user (instead of the usual single, unweighted value), we consider the feature vector to be the weighted sum of the vectors for each feature type appearing for that page, normalized by the sum of all the weights. So if a user had n weighted LDA entries (a_i, w_i) (where a_i being the topic index, and w_i being its weight), the vector for its LDA feature would be $\sum_{i=1}^{n} w_i \cdot p_{a_i} / \sum_{i=1}^{n} w_i$. The combination of the above two sets of features (user and page features) led to almost the best results we were able to get. An additional set of features we considered were ad features. The ad vector is composed as a normalized sum of vectors, one for each of the different possible ad features. The following ad features were used:

Ad ID - A unique ad identifier.

Campaign ID - An identifier for the campaign the ad is part of.

Advertiser ID - An identifier for the advertiser.

Several other approaches were tried, but without showing improvement in the results. We detail them below as we do consider them interesting on their own merit and they might be useful in other similar types of ad targeting problems.

We tried considering the history of the page. Here, statistical breakdowns of the age and gender (based on available coverage) of users landing on the page were generated for each page. Then, unknown users were considered to have a gender/age corresponding to that breakdown. So if a page had received 90 male events and 10 female events, each unknown user was considered to have a gender vector which was 0.9 times the male latent vector and 0.1 times the female latent vector. These vectors can then also be optimized using SGD for this event even though we didn't actually have the user information.

This approach, unfortunately, did not yield further improvements, though it is possible that this is due to the relative sparsity of relevant information in the available data. We do consider this also a viable approach to attempt on future instances of similar problems.

Another attempted approach was to consider features based on the content generators of the page, i.e., the users who asked or answered the question on that page. In this approach the data about the generator (or generators) of the content on the landed page is considered as part of the user features. This approach seems applicable to a wide range of similar problems, as blogging platforms, forums, photo hosting sites and many other popular web destinations. They are all faced with a similar situation, in which events are generated by unknown users in pages where the content was generated by a registered user, so data is available for the content generator, but not for the user generating the event.

The following information about the page's content generators was available, all at 100% coverage:

Asker's age: - The age of the question asker, was bucketed into a domain of size 12.

Asker's gender - Gender of question asker (male/female/unknown).

Answerer's gender - The genders of the answerers for that page.

Answerer's age - A list of the ages of the question answerers.

Features such as Answerer's age or gender, which hold multiple values per event, are handled in a similar way to LDA topics or the statistical breakdown described above. So, for instance, if the page had 8 male answerers and 3 female answerers, each user was considered to have an answerer gender vector which was $\frac{8}{11}$ times the male latent vector and $\frac{3}{11}$ times the female latent vector. While this approach did not improve our results, we do consider it as something to attempt in future problems of this type.

6. BASELINE ALGORITHMS

The following algorithms were considered as baseline algorithms:

Popularity: We score each ad based on its popularity, hence the pCTR of each ad is $\frac{\text{number of clicks}}{\text{number of impressions}}$. To compensate for variations in popularity over time, we apply an exponential decay function, i.e., we multiply the current totals by 0.9 after each time period.

Popularity per category (PPC): Here, we consider the popularity of an ad separately for each category (see category feature in section 5). Thus, the score of an ad on an event for a category is:

$$\frac{\text{number of clicks for the ad in this category}}{\text{number of events for the ad in this category}}$$

When little information is available for an ad in a particular category, its overall popularity is used instead.

7. EVALUATION

7.1 The Dataset

Our dataset is comprised of a *random sample* of event data logged from 12 days of traffic at Yahoo Answers, averaging 9.8 million impressions per day (varying from 7.5 to 10.9). The frequency of visits per user during the observed period of 12 days is depicted in Figure 1. During this time, over 77% of the observed events were of one-time visitors – these users were served only once during the observed time period.

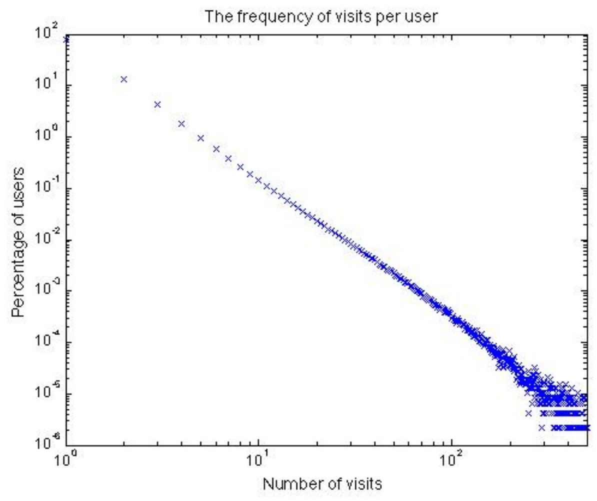

Figure 1: Visits frequencies per user. More than 77% of users has a single visit over the observed period of 12 days.

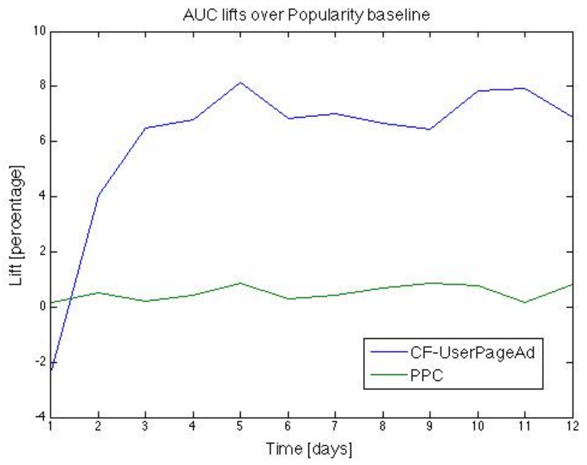

Figure 2: Daily AUC lifts of CF-UserPageAd and PPC baseline over the popularity baseline.

7.2 Experimental Setup

For each event in the dataset, the tested algorithm first provides a CTR prediction. Then, the event is used for training the tested algorithm, before continuing to the next event. At the end of each day of data, the algorithm's performance was evaluated for that day of data, based on the computed scores of that day. The algorithms' model was preserved from one day of data to the next.

7.3 Results

In this section we present the results of our testing. In each variant of our algorithms we consider the best performance achieved after parameter tuning. Each algorithm must handle the cold start problem as it trains a new model from scratch. Therefore, it can be noticed that our results fluctuate for the first few days before settling into clearly distinguished patterns. As such, we will only

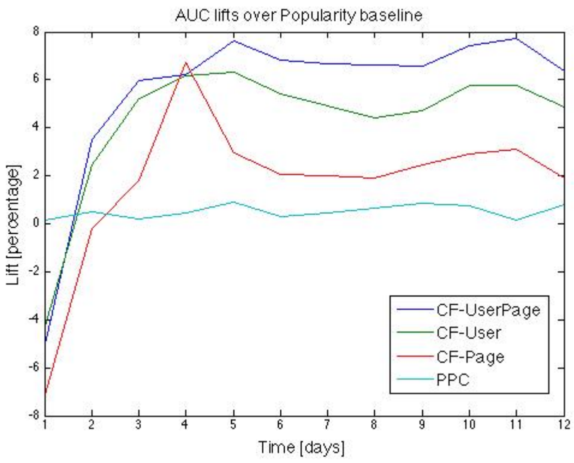

Figure 3: Daily AUC lifts of algorithm variants (user and page features based) and PPC baseline over the popularity baseline.

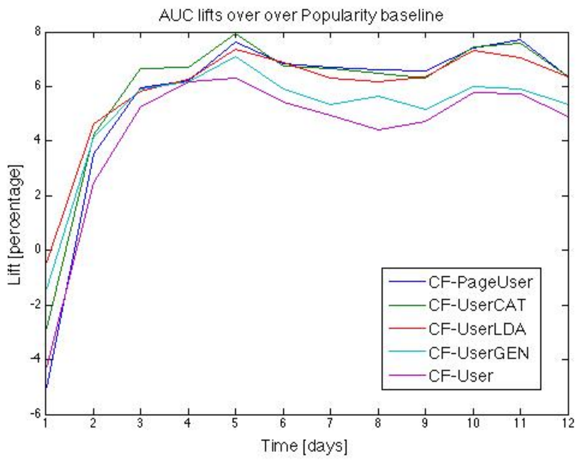

Figure 4: Daily AUC lifts of algorithm variants (incorporating various page features) over popularity baseline.

consider the results on the latter parts of the data.

As an overview of our results, in Figure 2 we first take a look at how the best performing instance of our algorithm compared with the baseline algorithms. The AUC lifts provided by the following algorithms over the popularity baseline are shown in this figure:
PPC - the popularity per category baseline algorithm.
CF-UserPageAd - our best performing configuration. This version of our algorithm combines user, page and ad features (as described in section 5), but doesn't use content generators based features or statistics gathered about the landed page.

We can see that CF-UserPageAd starts out worse than the popularity baseline, as cold start is more of a problem for it. However, it catches up and begins improving during the second day (after 12-15 million events) and by the third day it has settled into a consistent and significant 6-8% improvement over the popularity baseline, and nearly that much improvement over the PPC baseline (which only slightly outperforms popularity).

Next, we examine the contribution of each type of feature to CF-UserPageAd's improvement. As is soon to be shown, almost all of this improvement comes from the page and user features. We continue with an analysis of these.

Figure 3 depicts the temporal AUC lifts for the following configurations over the Popularity baseline:
CF-Page - CF algorithm using only the page features.
CF-User - CF algorithm using only the user features.
CF-UserPage - CF algorithm using both page and user features.
PPC - the popularity per category baseline algorithm.

Here, we can see that each type of feature contributes a significant portion of the algorithm's gains, but neither is enough to account for all of them. User features provide the more significant lift, despite the lack of what would traditionally be considered "strong" signals, while page features provide an additional, significant and consistent lift.

We also note that these lifts are nearly orthogonal, as the sum of the lifts from CF-User and CF-Page separately are only slightly larger than the total lift. And that each separately is already significantly better than baseline algorithms.

Next, we examine in Figure 4 how adding the different page features contributes to the improvement in performance. The curves depicted in this figure present the daily AUC lifts over the popularity baseline.

CF-User - CF algorithm using only the user features.
CF-UserLDA - CF algorithm using all user features as well as the content LDA topics.
CF-UserCAT - CF algorithm using all user features as well as the page category.
CF-UserGEN - CF algorithm using all user features as well as the content generator features.
CF-UserPage - CF algorithm using both page and user features.

We can first see that the content generators' features provide only a small lift, which can be considered evidence that in our setting, content generators may not provide a good model for content consumers. The lift provided by these feature is also completely contained in the lift provided by the page features (i.e., using content generator features on top of page features provides no lift).

Of the two page features, category and content LDA, there seems to be a slight edge in lift to the category (as the figure shows). Overall it seems neither gives a consistent boost over user features, and that the overlap between them is large. This might be as expected, as both come from attempts to catalog the text by subject, and both seem to have succeeded to similar degrees.

A small additional improvement is gained by incorporating ad features. This is demonstrated in Figure 5, where temporal AUC lifts of the following configurations over the popularity baseline are presented:
CF-UserPage - CF algorithm using both page and user features.
CF-UserPageAd - CF algorithm using page, user and ad features.

It can be seen that the improvement gained with ad features, though small, is consistent and therefore, noteworthy. Adding content generators features at this point have no effect.

8. CONCLUDING REMARKS

In this work we examined the problem of ad-serving with little or no access to user information. We considered approaches that combine textual analysis and content categorization of the page, information about the content generators for that page, statistical breakdowns of past known visitors to the page and available user information. Extensive testing was performed on a real instance of this problem, Yahoo Answers, which receives a large portion of its page-views from unknown users clicking through external search results. Results show that combining available user information with analysis of the page content yielded significant and consistent

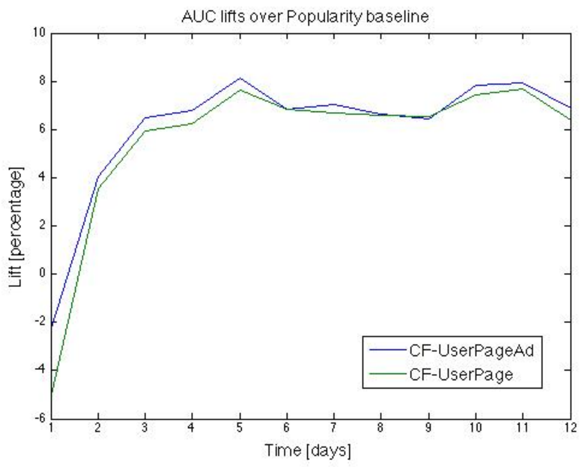

Figure 5: Daily AUC lifts of algorithm variants (incorporating various ad features) over popularity based baseline.

improvement over baseline algorithms (see CF-UserPageAd in section 7.3). Future work includes online evaluation of our methods on live traffic as well as experimenting with more elaborated models.

Acknowledgments

The authors would like to thank Yuval Pinter, Alex Nus, Dan Peleg, and Natalie Aizenberg for their help in generating the "Yahoo Answers" dataset used for the evaluation process.

9. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proc. SIGKDD'2009*, pages 19–28. ACM, 2009.

[2] M. Aharon, N. Aizenberg, E. Bortnikov, R. Lempel, R. Adadi, T. Benyamini, L. Levin, R. Roth, and O. Serfaty. OFF-set: one-pass factorization of feature sets for online recommendation in persistent cold start settings. In *Proc. RecSys'2013*.

[3] M. Aharon, A. Kagian, Y. Koren, and R. Lempel. Dynamic personalized recommendation of comment-eliciting stories. In *Proc. of RecSys'2012*.

[4] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. In *Proc. WWW'2012*.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Proc. NIPS'2011*.

[7] G. Dror, N. Koenigstein, and Y. Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item. In *Proc. RecSys'2011*.

[8] 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.

[9] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[10] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proc. WSDM'2011*.

[11] N. Golbandi, Y. Koren, and R. Lempel. On bootstrapping recommender systems. In *Proc. CIKM'2010*.

[12] A. Gunawardana and C. Meek. Tied Boltzmann machines for cold start recommendations. In *Proc. RecSys'2008*.

[13] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *Proc. RecSys'2009*.

[14] 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 *Proc. SIGKDD'2014*.

[15] A. Kohrs and B. Merialdo. Improving collaborative filtering for new users by smart object selection. In *Proc. ICMF'2001*.

[16] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. KDD'2008*.

[17] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[18] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[19] S.-L. Lee. Commodity recommendations of retail business based on decision tree induction. *Expert Systems with Applications*, 37(5):3685–3694, 2010.

[20] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[21] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proc. SIGKDD'2013*.

[22] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proc. RecSys'2009*.

[23] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup 2007*.

[24] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proc. International Conference on Intelligent User Interfaces*, 2002.

[25] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100, 2008.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW'2001*.

[27] I. Szpektor, Y. Maarek, and D. Pelleg. When relevance is not enough: promoting diversity and freshness in personalized question recommendation. In *Proc. WWW'2013*.

[28] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proc. SIGIR'2011*.