Predictive Client-side Profiles for Keyword Advertising

Mikhail Bilenko
Microsoft Research
Redmond, WA 98052, USA
mbilenko@microsoft.com

Matthew Richardson
Microsoft Research
Redmond, WA 98052, USA
mattri@microsoft.com

Abstract

Current approaches to personalizing online advertisements rely on estimating user preferences from server-side logs of accumulated user behavior. In this paper, we consider client-side ad personalization, where user-related information is allowed to be stored only within the user's control (e.g., in a browser cookie), enabling the user to view, edit or purge it at any time. In this setting, the ad personalization task is formulated as the problem of iteratively updating compact user profiles stored client-side to maximize expected utility gain. We compare the performance of client-side profiling to that of full-history server-side profiling in the context of keyword profiles used to trigger bid increments in search advertising. Experiments on real-world data demonstrate that predictive client-side profiles allow retaining a significant fraction of revenue gains due to personalization, while giving users full control of their data.

1 Introduction

Personalization has become a core component of modern web applications, where its uses vary from re-ranking search engine results to item recommendations in domains ranging from news to online shopping. Traditional uses of personalization center on customizing the output of information systems for each user based on attributes stored in their profile. Profile attributes may be explicitly or implicitly obtained, where explicit attributes are volunteered by the user or computed deterministically (e.g., user-submitted demographics, or IP-based location). Implicit user attributes are inferred based on logs of the user's prior behavior, e.g., of their past searching, browsing or shopping actions. A wide variety of personalization approaches have been proposed in recent years; notable examples include methods that leverage correlations between the behavior of multiple users (i.e., collaborative filtering), and approaches that use past behavior to assign users to one or more pre-defined categories (e.g., to behavioral targeting segments).

Raw behavior logs used to infer implicit user attributes are typically stored in the online service's datacenter (*server-side*), where they are processed to compute each user profile in a compact representation chosen for the application at hand. Examples of such representations include categories for behavioral targeting [3][21] and low-dimensional latent topics for collaborative filtering methods based on matrix decomposition [13]. The resulting profiles are used in subsequent interactions with the user to adjust the output of the application to user preferences.

Server-side aggregation is being increasingly questioned by consumer advocates due to the fact that it does not provide users the ability to view or control the data associated with them. As a result, there has been a rising interest in privacy-enhanced approaches to personalization, with one such approach being category-based profiles constructed and maintained entirely on the user's machine (*client-side*) for personalizing search results [18][20]. However, the trade-offs involved in moving user profiles client-side remain unclear.

In this paper, we formalize the problem of constructing client-side profiles based on the general framework of maximizing expected personalization utility. For many personalization utility functions that can be formulated as coverage problems, profile construction is a submodular optimization task, allowing efficient approximation algorithms with strong guarantees. We focus our attention on the utility of keyword profiles in search advertising accumulated via *bid increments*: keyword bid increases that allow advertisers to differentiate their campaigns for users with a strong interest in the topic. For this setting, we compare the performance of online, client-side profiling to full-history server-side profiling. Experiments on real-world data demonstrate that client-side profiling retains a significant fraction of server-side profiling revenue gains, while allowing users to opt out of server-side logging and gain full control of their behavioral history.

It is important to note that the presented approach is not a privacy-preserving ad delivery mechanism [7][6][19]: such mechanisms require installation of additional client software from users and significant changes to existing infrastructure and mechanisms from ad platforms. Our approach also does not aim to provide users statistical privacy guarantees such as those pursued by research in k-anonymity and differential privacy [14]. Instead, the goal of the paper is to describe a methodology for continuing to serve personalized advertising to users who have opted out of server-side logging, and to analyze the gap in personalization utility between client-side and server-side approaches in the context of search ads.

2 Advertising Personalization

Let $\mathcal{V}$ be the finite set of user behavior items, Φ be the domain of item descriptors, and $\mathcal{O} = \mathcal{V} \times \Phi$ be the domain of observed events. For example, in search advertising, $\mathcal{V}$ is the set of all advertiser-bid keywords that are matched to user queries, and $\Phi = \mathbb{R}^d$ contains vectors of features associated with a keyword being matched to a query. Then, every user query q can be represented as an observed event $o = (v, \phi)$ where $v \in \mathcal{V}$ is the most relevant ad keyword for the query, and $\phi \in \Phi$ is a vector of features capturing such event properties as the timestamp, keyword similarity to the query, user location, etc.

Let $\mathcal{H}$ be the domain of all sequences of observed events, and $\mathcal{P}$ be the domain of profile representations. A *profile construction* function $f: \mathcal{H} \to \mathcal{P}$ takes a sequence of events observed over a time period T, $H_T = \left(o_{t_1} .., o_{t_n}\right), \forall t_i \in T$, and produces a user profile $p \in \mathcal{P}$. This definition can be trivially extended to include explicit or time-independent attributes such as demographic data.

The objective of profile construction is to maximize some utility function that captures the increase in performance for the task at hand (the benefit of personalization). The utility function u is computed post-hoc by evaluating performance of the system over a future interval on a profile constructed during a preceding time interval, $u: \mathcal{H} \times \mathcal{P} \to \mathbb{R}$. The optimal profile construction function f^* maximizes expected utility: $f^* = \arg\max \mathbb{E}[u(H_{T^+}, f(H_{T^-}))]$, where the expectation is computed over the probability distribution of behavior across all users, while T^- and T^+ are time intervals over which the profile is constructed and used, respectively.

A number of utility functions have been considered in prior work on personalization, e.g., measuring improvements in web search result accuracy has been performed via gains in average precision [20] or click probability prediction [3][5][21]. The value of information approach [9] provides a general probabilistic framework for computing the utility of personalization.

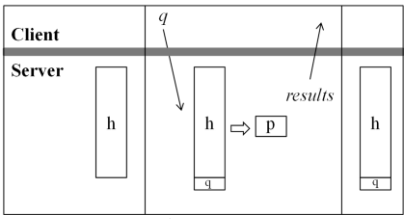
2.1 Keyword-based Profiles

Although the above formulation applies to arbitrary domains, we will now focus on the search advertising setting where both observed events $\mathcal{V}$ and profile representation $\mathcal{P}$ correspond to bidded keywords. Unlike display advertising, modern search and contextual ad platforms associate advertisements with bids on individual keywords, which are then matched against queries or page content (either exactly or approximately). Hence, user profiles comprised of keywords can be naturally integrated with existing advertising systems and campaigns. To be useful in advertisement selection, ranking and pricing, such profiles contain the keywords in which a user has shown historical, as well as predicted future interest. By allowing advertisers to target users based on their keyword profiles, pay-per-click campaigns can be refined for users for whom they are likely to be more ef-

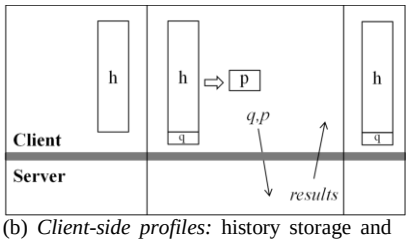
fective. For example, a diving equipment store may be willing to pay more for users who are predicted to have a long-term interest in specialized diving keywords, since they are more likely to purchase high-end items.

Requiring the ad keyword to be a part of a user's profile is limiting, since profile size is small, while the number of keywords that express any particular intent is large. E.g., an advertiser bidding on "*grass seed*" may wish to target users with corresponding long-term interests, but would then miss users whose profile contains "*lawn repair*". Note that the specificity of the interest rules out using lower-granularity approaches such as segments or categories. This problem is a key task for the ad selection phase of advertising delivery, where it is solved by constructing a weighted directed graph, G, with vertices representing all keywords and edges connecting keywords that should be matched. Building such graphs is a well-studied problem for online advertising [7][10][17], as it enables non-exact matching of keywords to queries (known as "broad" or "advanced" matching). Directed links allow e.g. "*Boston mortgage*" to point to "*mortgage*" to indicate that a user interested in the former will be interested in the latter, while the opposite is not generally true.

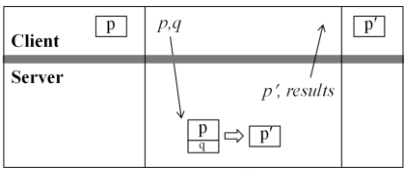
Given a user profile of k keywords, $P = \{w_1, .., w_k\}$, we consider a current context (query or webpage) keyword w to *match* the user profile if it is either contained in it, or is a direct neighbor of at least one of the profile keywords. Because utility is typically an additive function of individual context matches, finding the optimal profile is an instance of the maximum coverage problem: selecting a fixed-size set of profile keywords maximizing a set-based objective function. While the problem is NP-hard in general, personalization utility being submodular guarantees that the greedy algorithm of Nemhauser et al.[14] produces a $(1 - 1/e)$-approximate solution.



(a) *Server-side profiles*: The server stores the entire user history *h*, which is augmented with the query *q*, compacted into a user profile *p*, and used to serve results.



(b) *Client-side profiles:* history storage and compaction are performed on the client, with the profile sent to the server at delivery time.



(c) *Online client-side profiles*: The client stores only the compact profile, which is sent to the server along with the query. The server returns the updated profile and the results.

Figure 1: Comparison of server-side, client-side, and online client-side profiles.

3 Online Client-side Profiles

The general definition of profile construction in Section 2 presumes that the profile is constructed based on the complete history of user behavior over the previous time period, H_T−. Currently, this history is stored server-side and updated as necessary. To give users full control over their information, the history and profile must be stored on the client, potentially incurring prohibitive local storage costs. Profile construction would happen either on the client (requiring additional browser components to be installed to enable computation in the browser [7][6][19], presenting a significant barrier to wide adoption) or the server (requiring the history to be communicated to the server whenever the profile is to be updated, thus incurring significant communication overhead).

Given these concerns, we consider the scenario where the history is not stored, and the (relatively small) user profile is stored on the client. The compact profile is sent to the server along with the current context (query or webpage id). The profile is then utilized on the server, updated and returned along with the ads served. This scenario is supported by current web browsers natively via cookies.

Updating client-side profiles online changes the problem from one of constructing a profile given the full user history to one of revising the present profile based on the current context. The corresponding profile construction function for updating the profile on per-event basis is then defined as $f_{online}: \mathcal{O} \times \mathcal{P} \rightarrow \mathcal{P}$, with the profile update at i^{th} observation computed as $p_i = f_{online}(p_{i-1}, o_i)$.

The corresponding utility, u_{online}, aggregated over interval T^+ is then computed as $u_{online}(p, H_{T^+}) = \sum_{o_i \in H_{T^+}} u(p_{i-1}, o_i)$ where p_o is the initial empty profile.

To alleviate the myopia suffered by the incremental update, we augment the profile with a cache of *m* recently seen keywords which are not a part of the profile, yet are retained alongside it client-side. Selecting the optimal *k* profile keywords from the up to *k+m+1* known keywords is still submodular and may still be approximated using the greedy algorithm.

4 Method

We apply online client profiles to the *bid increment* setting, where advertisers are given the option to have their bids increased for users that have a particular profile attribute. Conceptually, the bid increment indicates expectation of the ad being more effective when shown to or clicked by such users. Bid increments are already commonplace in display advertising platforms, where they are based on either demographic attributes, or broad, loosely defined categories [21]. In keyword advertising, the bid increment is charged if the advertising keyword matches those in user's profile. This notion integrates naturally with existing keyword campaigns in search and contextual advertising. The corresponding utility function is the total bid increment amount for clicks in matched settings. Note that this utility function underestimates the actual increase in revenue, as it does not account for revenue increases resulting from improved re-ordering of the ads due to bid increment. Because a higher bid can only increase the ad's position in a second-price auction, it makes corresponding clicks more likely, hence amplifying the gains. The utility formulation also does not account for expected gains in revenue due to cost-per-click increases for non-matched ads.

4.1 Utility Computation

Estimating the bid increment utility of a set of keywords requires computing the probability that, in the next time step, the user will click on an ad for one of those keywords or their graph neighbors. This is done by estimating the expected future clicks individually for each keyword involved, then combining these (according to the keyword graph) to compute overall utility.

We employ a machine learning approach for this estimation: a parameterized function is trained on historical data to minimize the difference between observed clicks and those predicted by the function, based on a set of features. Historical data comes from users who do not opt-out of server-side profiling (some number of such users can always be retained via incentives). The features are functions of the keyword, context and/or user that assist the model in predicting whether the user will click on an ad for the keyword in the future. Our model incorporates three feature sets:

- ***User Prior Features*** based on the counts of past searches and ad clicks for the user, which can be stored alongside the cache and profile client-side and incremented continuously;
- ***Recency Features*** based on recency of past occurrences for each considered profile/cache keyword, captured via 10 geometrically increasing lookback windows;
- ***Time-weighted Frequency Features*** based on heuristic time-decay functions that assume that the probability of a future click decreases with time, yet increases with the count of past occurrences.

Logistic regression based on the L-BFGS optimizer was chosen as the learning algorithm for utility prediction as it outperformed a number of other learners in pilot studies. Once the utility prediction function is trained, online profile construction is performed by considering every keyword (profile, cache, and context, including search queries and advertisements shown) and their children as candidates for inclusion in the profile, using predictions of their expected clicks in the subsequent time interval. Iteratively, the keyword with highest incremental utility is added to the profile, where incremental utility is the sum of the keyword's and its neighbors' expected clicks, subtracting those already covered by keywords selected in earlier iterations. Because the utility function is submodular and monotone, this algorithm is guaranteed to find a profile with utility that is at least $1 - 1/e = 63\%$ of optimal.

5 Results

Experimental evaluation of the proposed approach for constructing compact keyword profiles was performed using advertising system logs from the Bing search engine over a two-week period. The first week's data was used for training of the utility predictor as described in Section 4. The training set contains the candidate keywords and their features, extracted over the week-long period, with labels (clicks) obtained from the subsequent one-day interval. Although this reduction of utility prediction to a standard supervised learning task neglects the online setting (i.e., the pool of candidates during online client-side construction is significantly smaller), it provides a reasonable batch-setting approximation, leaving truly online approaches as an interesting direction for future work. The experiments relied on the keyword graph used for matching related keywords in production.

With the utility predictor trained on past data, we evaluated the efficacy of online client-side profiling on a held-out set of over 20,000 users during the subsequent week. Profiles constructed in the server-side setting (using the complete user behavior history) were compared to those constructed in the online client-side setting (using a small cache of user behavior). Figure 2 illustrates the relative performance of client-side personalization with respect to server-side personalization for different profile sizes, demonstrating that even modest cache sizes provide performance that is comparable to server-side profiling with complete history. Indeed, for a profile size of 20 and a cache size of 40, online client-side profiling captures 98% of the revenue gain of server-side profiles, while giving full control of data and privacy to the user. Such settings are reasonable as they allow fitting both the profile and the cache with corresponding features into the cookie size limit of 4 kilobytes.

Figure 3 compares the performance of a sample client-side profiling setting to that of server-side profiling, and also to the maximum achievable performance. The latter corresponds to an oracle selecting the optimum profile from the user's complete history of past behavior to obtain maximum future utility presciently, thus bounding the amount of improvement that could be obtained with more sophisticated features or learners.

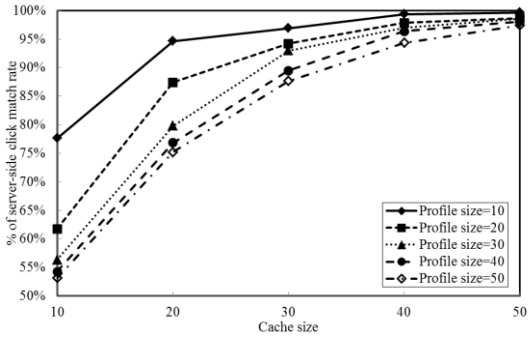


Figure 2: Client-side profiling accuracy for different profile sizes

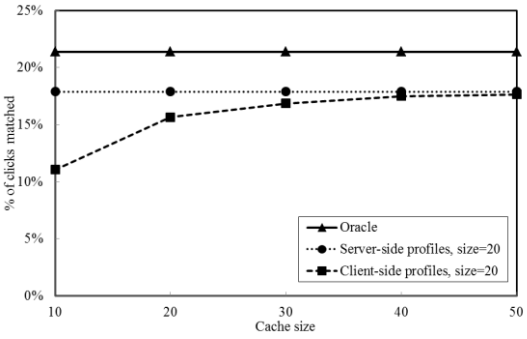


Figure 3: Comparison of client-side, server-side and maximum achievable accuracy

As these results demonstrate, our overall approach to constructing keyword profiles achieves a significant fraction of the maximum possible performance (83% of the oracle, for profiles of size 20). The oracle's upper bound of 22% of matched clicks captures the low overall predictability of future ad clicks. Finally, we note that if advertisers opted for a 25% bid increment (an average of comparable increments seen in existing targeting experiments), client-side keyword profiling would increase overall search engine revenue by over 6%, a sizeable gain that can be realized in a privacy-friendly way.

6 Related Work and Conclusions

Previous work on profile construction for ad personalization has focused on display advertising, where profiles consist of high-level categories also known as behavioral targeting segments [3]. As an alternative to predefined segments, Yan et al. [21] evaluated whether clustering methods to identify groups of users that show similar CTR behavior. Le et al. [12] also investigated clustering users

based on their browsing behavior, observing that users who visit the same web pages have similar interests and thus click on similar ads.

Personalization for search advertising was previously considered only in the context of modifying clickthrough prediction estimates. Chen et al. [4] propose a latent-topic model for user-dependent CTR prediction, where each user is represented by a mixture of automatically derived topics. Similarly to other work on behavioral targeting to date, latent topics provide a relatively low targeting granularity, and are also not as transparent as keyword based profiles. Recent work on user-dependent CTR prediction for sponsored search by Cheng and Cantú-Paz [5] used two types of user features: demographic categories, and features derived from the user's historical CTR behavior for a given advertiser or query, presuming complete server-side history. Compared to that work, keyword profiles provide orthogonal benefits of making personalization explicit to advertisers via bid-increments, while allowing for user transparency and client-side storage. Combining prior work on personalizing CTR prediction with keyword profiles and deriving advertiser-side mechanisms for pricing increments are two interesting directions for future work.

In contrast to ad personalization, search result personalization has been a topic of active research for many years. Kelly and Teevan [11] provide a survey of techniques that build profiles of users based on their past behavior, using a variety of signals that include query history [16], browsing activity [13], or a combination of the two [1][18].

To our knowledge, this work is first to consider the problem of online construction of client-side keyword user profiles. Our framework allows making profiling transparent to users with little storage or communication overhead. Initial results demonstrate that maintaining client-side profiles incrementally through caching and greedy optimization enables ad platforms to allow users to opt-out of server-side logging without significant losses in revenue from personalization based on complete user history. While a number of interesting research challenges remain in developing better online learning algorithms for this problem, we believe our general approach has significant potential for improving personalized advertising.

References

[1] M. Bilenko, R. White, M. Richardson, G.C. Murray. Talking the talk vs. walking the walk: salience of information needs in querying vs. browsing. *SIGIR-2008*.

[2] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. *SIGIR-2007*.

[3] Y. Chen, D. Pavlov, J. Canny. Large-scale behavioral targeting. *KDD-2009*.

[4] Y. Chen, M. Kapralov, D. Pavlov, J. Canny. Factor modeling for advertisement targeting. *NIPS-2009*.

[5] H. Cheng, E. Cantú-Paz. Personalized click prediction in sponsored search. *WSDM-2010*.

[6] M. Fredrikson and B. Lifshits. REPRIV: Re-Envisioning In-Browser Privacy. MSR Tech. Report, 2010.

[7] S. Guha *et al.* Serving Ads from localhost for Performance, Privacy, and Profit. *HotNets 2009*.

[8] S. Gupta, M. Bilenko, M. Richardson. Catching the drift: Learning broad matches from clickthrough data. *KDD 2009*.

[9] D. Heckerman, E. Horvitz and B. Middleton. An Approximate Nonmyopic Computation for Value of Information. *IEEE PAMI* 15: 292-298, 1993.

[10] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. *WWW 2006*.

[11] D. Kelly and J. Teevan. Implicit feedback for inferring user preference. *SIGIR Forum, 2003*.

[12] T. Li *et al*. A Markov chain model for integrating behavioral targeting into contextual advertising. *Ad-KDD 2009 Workshop*.

[13] B. Marlin. Modeling user rating profiles for collaborative filtering. *NIPS 2004*.

[14] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. *KDD 2009.*

[15] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming, 14:265-294*, 1978.

[16] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. *WWW 2006*.

[17] F. Radlinski *et al*. Optimizing relevance and revenue in ad search: a query substitution approach. *SIGIR 2008*.

[18] J. Teevan, S.T. Dumais, E. Horvitz. Personalizing search via automated analysis of interests and activities. *SIGIR 2005*.

[19] V. Toubiana *et al*. Adnostic: Privacy Preserving Targeted Advertising. *NDSS 2010*.

[20] Y. Xu, B. Zhang, Z. Chen, K. Wang. Privacy-Enhancing Personalized Web Search. *WWW 2007*.

[21] J. Yan *et al*. How much can behavioral targeting help online advertising? *WWW 2009*.