Mining Advertiser-specific User Behavior Using Adfactors

Nikolay Archak
New York University, Leonard N. Stern School of Business
44 West 4th Street, Suite 8-185
New York, NY, 10012
narchak@stern.nyu.edu

Vahab S. Mirrokni
Google Research
76 9th Ave
New York, NY 10011
mirrokni@google.com

S. Muthukrishnan
Google Research
76 9th Ave
New York, NY 10011
muthu@google.com

ABSTRACT

Consider an online ad campaign run by an advertiser. The ad serving companies that handle such campaigns record users' behavior that leads to impressions of campaign ads, as well as users' responses to such impressions. This is summarized and reported to the advertisers to help them evaluate the performance of their campaigns and make better budget allocation decisions.

The most popular reporting statistics are the click-through rate and the conversion rate. While these are indicative of the effectiveness of an ad campaign, the advertisers often seek to understand more sophisticated long-term effects of their ads on the brand awareness and the user behavior that leads to the conversion, thus creating a need for the reporting measures that can capture both the duration and the frequency of the *pathways* to user conversions.

In this paper, we propose an alternative data mining framework for analyzing user-level advertising data. In the aggregation step, we compress individual user histories into a graph structure, called the *adgraph*, representing local correlations between ad events. For the reporting step, we introduce several scoring rules, called the *adfactors* (AF), that can capture global role of ads and ad paths in the adgraph, in particular, the structural correlation between an ad impression and the user conversion. We present scalable local algorithms for computing the adfactors; all algorithms were implemented using the MapReduce programming model and the Pregel framework.

Using an anonymous user-level dataset of sponsored search campaigns for eight different advertisers, we evaluate our framework with different adgraphs and adfactors in terms of their statistical fit to the data, and show its value for mining the long-term behavioral patterns in the advertising data.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Performance, Measurement, Economics

Keywords

sponsored search, ad auctions, online advertising, PageRank, user behavior models, clickthrough rate, conversion rate

1. INTRODUCTION

The Internet has become a major advertising medium. Although a number of different factors contributed to this, what distinguishes the Internet advertising from the offline advertising competitors is its inherently interactive nature. Measuring effectiveness of a particular advertising campaign and allocating the advertising budget optimally was and still remains a very challenging task, yet the Internet made the task easier by connecting ad impressions [1] to tangible user actions and artifacts such as posing a search query, clicking on an ad or converting [2]. The simplicity of measuring and attributing user clicks has established the clickthrough rate (CTR) [3] as the current de-facto standard of ad quality for sponsored search. It is now customary to define the advertiser's optimization problem as maximization of the expected number of ad clicks given a certain budget constraint [13, 23, 10]. The conversion rate (CR), defined similarly as the probability of the user conversion, is another popular ad effectiveness measure; together with the CTR it is frequently used by the advertisers to measure the return on investment of specific keywords in the advertising campaign.

Recent empirical studies show that the effects of online ads cannot be fully captured by the CTR or the CR. In particular, the sponsored search advertising, as well as the display advertising, can have a significant number of indirect effects such as building the brand awareness [18]. For instance, Lewis and Reiley [20], in cooperation between Yahoo! and a major retailer, performed a randomized controlled experiment to measure the effect of the online advertising on sales. They found that the online advertising campaign had substantial impact not only on the users who clicked on the ads but also on those who merely viewed them. In another study, comScore [8] reported an incremental "lift" of 27% in

[1]Impression is simply an act of showing an ad in a page viewed by the user.
[2]Conversions are defined by the advertisers and may represent a wide range of user activities from buying an item to simply spending a suitable amount of time on the advertiser's website.
[3]The clickthrough rate (CTR) is defined as the fraction of times an ad was clicked by users when shown.

the online sales after the initial exposure to an online ad, as well as lift in other important online behaviors, such as the brand site visitation and the trademark searches.

The advertisers seek to understand the impact of their ad not just on the immediate click or conversion, but the likelihood of the eventual conversion in the long term and other long term effects. Users take specific trajectories in terms of the search queries they pose and the websites they browse, and this affects the sequence of ads they see; conversely, the sequence of ads they see affects their search and browsing behavior. This interdependence results in structural patterns in users' behavior; advertisers need new tools and concepts beyond simple aggregates (like the CTR and the CR) to understand them.

What are the systematic ways to help the advertisers reason about structural correlations in the data? In this paper, we take an advertiser-centric data mining approach. We start with data that is directly pertinent to the advertiser's campaign, that is, user trajectories that involved ads from the campaign of that advertiser, including ad impressions, clicks and conversions. Such data can usually be reported to the advertiser, provided it is aggregated and anonymized appropriately. Next, we build a data mining framework that can help advertisers identify structural patterns in this data. Our contributions are as follows.

- We propose a graphical model based approach. We formulate graphs from the data called *adgraphs* to capture co-occurrences of events adjacent to each other in users' trajectories. Then, we introduce a variety of *adfactors*, where every adfactor is a scoring rule for nodes in the adgraph: these are designed to capture impact of ad nodes on eventual conversion. For example, we introduce adfactors based on random walks, that, for every event, calculate the long term probability that a certain random walk involving that event would eventually lead to conversion. Our paper presents highly efficient algorithms for constructing adgraphs and computing all adfactors we introduce. All algorithms were implemented using MapReduce [11] parallel programming model.

- Using data from the sponsored search campaigns of eight different advertisers, we study various adgraphs and adfactors. We validate the adgraph models by showing their statistical fit to the data. Also, we show interesting empirical properties of the adfactors that provide insights into user behavior with respect to brand vs non-brand ads. Moreover, we show various natural data mining queries on adgraphs and adfactors that maybe of independent interest to advertisers. Finally, using adfactors, we show how to efficiently prune the adgraph to localize and depict the influence of any particular ad by its small neighborhood in the ad graph.

Our approach works by transforming the dataset of users' trajectories into graphs. This is achieved by pooling data from different users. As a result, adgraphs lose information about specific users or their trajectories, and only encode aggregate information. Also, because of the way adgraphs pool data only based on adjacent events, they encode certain independence assumption about user behavior over paths of multiple edges. We carefully study statistical fit of adgraph models to the data to ensure that this assumption is reasonable. Pooling this way also results in significant compression. For large advertisers in sponsored search, the number of different queries the advertiser can be matched with is often in tens of thousands, and the number of viewing users can be in millions. In contrast, adgraphs have more manageable sizes.

Our approach generalizes to different settings. While in this paper we apply it to study user conversions in the sponsored search data, it can equally be used to study trajectories in sponsored search that lead to clicks only, or conversions when both sponsored search and content-based display ad data is available, etc.

2. RELATED WORK

There are empirical studies showing that the sponsored search advertising, as well as the display advertising, can have a significant number of the indirect effects such as building the brand awareness and lift in the cross-channel conversions [18, 20]. The prior research attempted to measure impact of an ad on the user conversion by the number of independent paths from the ad to the conversion event [5]. Our work here is a substantial generalization of the prior research, as we apply more advanced PageRank-based measures for analyzing pathways to the user conversion and, more importantly, introduce a generic adgraph/adfactor based framework for reasoning about the structural properties of the users' behaviors.

Mining patterns in the user behavior is not a novel idea. Market basket analysis using association rule mining [2] is a popular tool used by retailers to discover actionable business intelligence from user level transaction data. User behavior data has numerous applications, including, but not limited to, improving the web search ranking [1], fraud detection [12] and personalization of the user search experience [25]. In this paper, we mine the user behavior from the perspective of an advertiser running a sponsored search campaign.

There are several standard mining techniques that can be applied to the user-level advertising data such as mining for the frequent itemsets [2] or the frequent episodes in sequences of user actions [22]. However, these do not capture the structural correlations in the data such as the impact of multiple paths between events that our work captures. A different approach that attempts to capture structural correlations is mining the frequent substructures in graph data [16]. Due to data pooling in our graph construction, patterns that we identify may or may not have high support, because some patterns may combine behavior of multiple users. Also, we are not interested simply in frequent patterns but in patterns that indicate a high likelihood of the designated action: user conversion. Further, our approach of using the steady state probabilities of different random walks to capture structural correlations differs from the prior graph mining techniques. This approach is widely used in other areas including web search [24, 4], personalized video recommendations [6] and user signatures in communication graphs [9].

Our research is closely related to the problem of modeling user search sessions, for which a wide number of solutions have been proposed in the literature, including advanced latent state models such as vl-HMM [7] and Markov models that take into account transition time between user's actions [14]. We intentionally refrained from using complex generative models for the underlying data due to several reasons. At first, generative models often require individual user sessions for training, while, due to privacy reasons, advertisers usually have access only to aggregate level data. At second, advertisers generally do not have access to information on user actions for which the advertiser's ad was not shown (even at the aggregate level). Finally, the patterns of user behavior extracted from the data must be easy to communicate and represent, ideally in a graphical form. This is often not true for generative statistical models.

3. DATASET

Our sample covers 8 different anonymous advertisers, who were running a sponsored search campaign with Google. The dataset was collected at user level and includes information on a random sample of users who "converted" with the advertiser within a certain time period of several weeks. The activity of every user was tracked by a cookie, thus any user who deleted the cookie was later identified as a different user. For every (anonymous) user, the data has the information on all actions of this user before the conversion, where we define an action as either a search query issued on Google *and for which the advertiser's ad was shown in the paid search results* or a click on the advertiser's ad.

In the rest of the paper, we will refer to the first action type as an "impression" event (as it resulted in the advertiser's ad impression) and to the second action type as a "click" event. We emphasize that search queries for which the advertiser's ad was not shown, such as irrelevant search queries, queries on which the advertiser bid too low, or queries for which the advertiser was excluded from the auction due to a daily budget constraint, are *not* included in our dataset. While such data might be available in some form to the search engine, it is not reported to the advertisers, e.g. due to several privacy issues, and therefore we intentionally refrain from including it as an input into our data mining process. Same applies to the ad position and the competitors' information in the sponsored search auction: advertisers do not observe their ad placement and competitors' ads on an individual query basis. The information that we assume to be available for every event includes only the event timestamp, the search query issued by the user and the match type (exact or broad [4]).

The number of data points for the converted users varied from approximately 30,000 impressions and 10,000 clicks for the smallest advertiser to about 5.8 million impressions and 2.6 million clicks for the largest advertiser. [5] The number of different user queries in the data varied from approximately 500 for the smallest advertiser to about 27,000 for the largest advertiser.

A special event in our dataset is a user conversion. User conversions were reported by the advertisers, therefore the exact definition of what constitutes the conversion event is advertiser-specific. In practice, it can vary from visiting a particular web site page or registering on the website to making an expensive purchase online. We will use the term "conversion path" to represent an ordered sequence of events for a single user that ends in a conversion.

Finally, our dataset also includes data on users who were exposed to at least one ad impression of the advertiser but did not convert within the monitored time period. As the number of non-converted (yet) users is much larger than the number of converted users, we sampled from the pool of such users randomly with a sampling probability of 1%.

[4]Most search engines support broad match functionality, which allows for an imprecise match between a query issued by the user and the keyword the advertiser bids on [10].

[5]Note that the high clickthrough rate implied by these numbers is due to the fact that we describe the sample of users who eventually converted with the advertiser.

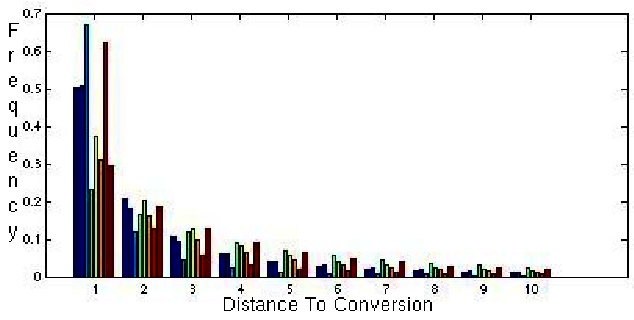

Figure 1: Distance to conversion

3.1 Descriptive Statistics

To ensure anonymity of the advertisers in our dataset, we do not report the descriptive statistics table, instead presenting two plots illustrating some of the most interesting properties of the data. We first ask how long (how many searches) does is take for a user to convert with the advertiser (assuming that the user eventually converts)? The exact value of the statistics depends on how we average data across different users. Figure 1 shows the histogram of the distance to conversion for a randomly selected *event* in the sample of converted users. The distance is defined as the number of future ad impressions for the same advertiser (including the current impression) that this user will get before the conversion. The data on the plot is slightly distorted (by a small multiplicative random factor for every advertiser) to preserve advertiser anonymity; the distortion does not affect the main message of the plot. In about one third of the cases, the user who converts will convert immediately after the observed impression, however a larger fraction (2/3) will be matched to the same advertiser at least one more time before they convert. Moreover, the distribution has a heavy tail: in about 10% of the observations, the user will be matched to the same advertiser in at least 10 more searches before eventually converting. The plot also shows that there is a lot of variation across advertisers, therefore our estimates of the distance to conversion have limited generalizability outside of our sample. What we would expect to generalize is the observation that *the interaction between the advertiser and the user is far more complex than a simple "search; click; convert" scenario and many users will be exposed to the advertiser's ad multiple times before they eventually convert.*

Finally, Figure 2 presents a plot showing the number of different path types of certain length in our conversion data. The path type is defined as the equivalence class of all conversion paths that are identical once the timestamps are omitted. Furthermore, we normalize all values by the number of different path types of the length 1, which is essentially the number of different user queries that we observed in the conversion data. In order to understand the plot better, one can think of the two extreme cases of the data generating process. In the first scenario, the data generating process samples new events i.i.d., i.e., the queries that the user issued before do not affect the queries that the user will use later. In such setting, the number of different path types of the length k (assuming a very large sample size) will grow

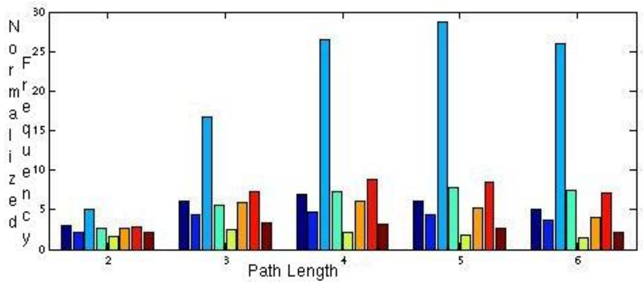

Figure 2: $\frac{\textbf{Number of different paths of length } k}{\textbf{Number of different paths of length } 1}$

exponentially with respect to the number of possible user queries (which is often in the order of thousands or even tens of thousands for a large advertiser). This is clearly not the behavior we see in Figure 2, even taking into account finite size of our sample (we formally test the i.i.d. model in the next Section). On the other hand, a perfect correlation between user searches would imply a flat plot, which is also not the case in our sample. While there is again significant variation across advertisers, we would expect the following observation to generalize: *there are strong "structural correlations" between user searches in a single conversion path.*

4. GENERATIVE MODELS

In this section, we formally define and evaluate a set of generative statistical models for the conversion path data. The first baseline model we evaluate is a simple *i.i.d. model*: every new event in a conversion path is sampled independently from everything that has happened before that. The number of parameters in the model is equal to the number of different user queries in the dataset (minus one).

All others models we evaluate represent user behavior as a random (Markov) walk in some directed graph (*adgraph*), what makes the models differ is the way the *adgraph* is constructed. The *adgraph* construction involves defining the set of graph nodes V, defining the set of graph edges E, and defining the edge weights w_{ij} which represent transition probabilities between nodes i and j. While definition of the nodes in the adgraph is model-specific, all adgraphs that we construct will have three special nodes: the "begin" node representing the start of the conversion (or non-conversion) path for any user, the "conversion" node representing the conversion event and the "null" node representing the state of the users who have not converted within the observation period. The "null" node is always an absorbing node and the "conversion" node always leads to the "null" node. The "begin" node has no incoming edges.

Simple Markov. In a "Simple Markov" adgraph, every graph node is an event (impression or click) and the edge weights are just probabilities of observing the two events consecutively in a conversion path. We also consider "Forward Markov" and "Backward Markov" adgraphs, in which we incorporate the distance from the first user observation (the distance to the last user observation) as a part of the node id, thus allowing the next event to be sampled depending on how many events for the same user we have observed so far ("Forward Markov") or how many more events we are going to observe before the user converts ("Backward Markov").

Regime Switching Markov. We also consider three additional "Regime Switching Markov" adgraphs. In all three models, the user is assumed to have two different possible regimes: the "search" regime representing a regular browsing behavior and the "interested" regime representing the behavior of a user interested in a particular advertiser. Thus, for every user query we will have two different nodes: one for the "search" regime and one for the "interested" regime. The user always starts in the "search" regime and, once switched to the "interested" regime, always stays there. We consider three possible signals of the user switching the "regime": clicking on the advertiser's ad ("RS Markov 1"), using the advertiser's brand name in a search query ("RS Markov 2") or a combination of both ("RS Markov 3").

Model	Stab. (RMSE)	Stab. (MAE)
i.i.d.	0.0000	0.0000
Simple Markov	0.1164	0.0414
Forward Markov	0.2128	0.1080
Backward Markov	0.2130	0.1068
RS Markov 1	0.1317	0.0489
RS Markov 2	0.1291	0.0475
RS Markov 3	0.1326	0.0494

Table 1: Model stability, out of sample

Model	-2 LogL (Out)	BIC (In)	AIC (In)
i.i.d.	8.9860	9.0042	8.9714
Simple Markov	5.1155	5.2718	5.0762
Forward Markov	4.3607	4.7591	4.3229
Backward Markov	3.6291	4.0926	3.5964
RS Markov 1	4.9965	5.1797	4.9545
RS Markov 2	5.0360	5.2146	4.9949
RS Markov 3	5.0169	5.1969	4.9759

Table 2: Model fitness, in and out of sample

We have evaluated all seven models in terms of several criteria: average stability (sampling variance) of the edge weights in the constructed graph (parameter estimates for the i.i.d. model) and fitness both in and out of sample. Stability scores were calculated in two forms: the average (across all edges) root mean squared error (RMSE) and the mean absolute error (MAE) of the edge weight estimates in a five sample split of the data. Table 1 shows that, with exception of the i.i.d. model, the edge weight estimates are quite volatile in our dataset, due to the fact that we have limited number of observations for many node pairs. [6] As the next Section shows, this in fact is not a problem for stability of the corresponding adfactors.

Next, we present the fitness results for all models in Table 2. We report the statistical model fit to the data in-sample as measured by the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) [3]. We also report the out of sample fit from five-fold cross-validation [19] using the log-likelihood as the fitness criteria. As expected, the i.i.d. model shows the worst fit by far. The best fitting model according to all three criteria is the "Backward Markov" model, indicating that the distribution of queries issued by the user varies significantly with the distance from the conversion event. Regime switching models fit slightly better than the "Simple Markov" model.

5. AD FACTORS

In this section, we define and evaluate a number of adfactors that capture structural correlations in the conversion path data. All adfactors are calculated directly on the graph structures we defined in the previous section, for instance, the same adfactor can be calculated both for a "Sim-

[6]While the i.i.d. model is stable, it provides bad fit to the data as shown in Table 2. This is just another example of the classic bias-variance tradeoff.

ple Markov" adgraph as well as for a regime switching adgraph. We present several alternative versions of the adfactors and evaluate their properties on the actual data. The adfactor is calculated for every node in the adgraph and, informally, can be thought of as a measure of structural correlation between this node and the target node (the conversion node). The simplest adfactor would be a simple edge weight between the pair of nodes in the adgraph. Such adfactor would capture a simple correlation between a pair of consecutive events: how often do users who get an ad impression for a particular query click on it? (clickthrough rate) how often do users who click on the ad convert with the advertiser? (conversion rate) The more sophisticated adfactors we define are expected to capture the structural properties of the adgraph such as: ceteris paribus, the larger the number of the disjoint paths from an event to the conversion is, the higher the adfactor should be, and also, the closer these events are, the higher the adfactor should be.

Three important points should be made about the adfactors. First, while we expect the adfactors to have useful applications (and we show some of the applications in this paper) and be useful for ranking and reporting purposes, we do not and cannot attach any causal meaning to the values. What adfactors capture are correlations not causations. This is even true for the simplest of the currently used models of ad performance such as the conversion rate; what the conversion rate says is what fraction of the users converted with the advertiser after clicking on an ad, not what fraction of the users converted with the advertiser because of the ad (some users would have converted anyway even if the ad was not shown; other users who did not convert immediately may have converted later because of the ad). Establishing causation would require running an actual experiment or, at least, having a good exogenous source of randomness in the data; this is beyond the scope of the paper.

Second, there is no ex-ante measure of quality of an adfactor and we are not going to say that one adfactor is always better than another. What we will say in the rest of the paper is what structural properties of the data the different adfactors are able to capture. Third, in order for the adfactors to be practically useful, they should be efficiently computable on large-scale data sets. To address this issue, we design efficient algorithms for all adfactors we introduce in this paper, using distributed computation paradigm. In fact, for one of the adfactors (the PPR adfactor), we can directly use a local pushback algorithm developed by Andersen et. al [4]. For the rest of adfactors, we present suitable adaptations of the pushback algorithm.

5.1 LastAd

LastAd adfactor represents the conversion rate of the corresponding ad: it measures the weight of the direct edge from the corresponding node to the conversion node. Formally, for any event e (either simple ad impression or ad click), the LastAd adfactor is

$$\text{lastad(e)} = \frac{\text{number of occurrences of } \{\ e, \text{conversion}\ \}}{\text{number of occurrences of } e}$$

5.2 PageRank contribution (PPR, CPR)

PageRank contribution adfactor is defined as the Personalized PageRank contribution of the evaluated node v to the conversion node c ($\text{ppr}(v, c)$). Below, we give a formal definition.

Assume some restart probability $\alpha > 0$ and a graph given by the $n \times n$ random walk matrix M ($M(i,j) = \frac{w_{ij}}{\sum_k w_{ik}}$). Let I be the identity matrix and $\mathbf{e}_v$ is the row unit vector whose v-th entry is equal to one.

The Personalized PageRank matrix is defined as a n by n matrix solution of the following equation

$$\text{ppr}_\alpha = \alpha I + (1-\alpha)\,\text{ppr}_\alpha\, M.$$

Fix a node v. The v-th row of the matrix satisfies

$$\text{ppr}_\alpha(v,\cdot) = \alpha \mathbf{e}_v + (1-\alpha)\,\text{ppr}_\alpha(v,\cdot)M.$$

This is known as the Personalized PageRank (PPR) vector of a node v with restart probability α and it was introduced by Haveliwala [15] as the stationary distribution of a random walk on the graph with α probability restart at node v after each step.

The dual construct to the PPR vector is obtained by taking a single *column* of the matrix. The column corresponding to the conversion node c, consists of all PPR contributions for different nodes v and the fixed conversion node c and, when written in a row form, solves the following equation

$$\text{ppr}_\alpha(\cdot,c) = \alpha \mathbf{e}_c + (1-\alpha)\,\text{ppr}_\alpha(\cdot,c)M^T.$$

This is PageRank contribution (CPR) vector and it is denoted as $\text{cpr} \equiv \text{ppr}_\alpha(\cdot,c)$ [4]. The CPR vector naturally captures important structural properties of the graph such as the number of different paths from every node v to the conversion node c as well as the lengths of these paths. A nice property of cpr is that the sum of all cpr's for the node c is equal to the PageRank of the node c, so it can be thought of as a way to divide the PageRank score (which captures the total credit of the conversion node in the graph) into contributions from other nodes. Moreover, varying the restart probability of the random walk (α) can be used to fine-tune the trade-off between the short-term and the long-term effects of the ads. The higher the restart probability is, the more the score captures the short-term contribution of the ad as compared to the long-term one.

5.3 Eventual Conversion

The *eventual conversion* adfactor is *a limiting case of* cpr *when the restart probability α converges to zero.* The following proposition says that, after normalization, this adfactor is equivalent to the probability of hitting the conversion node as opposed to hitting the null node in a random walk with no restart. This adfactor is denoted $\text{hit}(\cdot,c)$.

PROPOSITION 1. *Let $\hat{M}$ be the random walk matrix with the "null" node excluded and assume that the matrix $I - \hat{M}$ is invertible, where I is the identity matrix.* [7] *Define the "hit" vector* $\text{hit}(\cdot,c)$ *as a row vector solving the following equation:*

$$\text{hit}(\cdot,c) = \text{hit}(\cdot,c)\hat{M}^T + e_c.$$

Then,

$$\lim_{\alpha\to 0} \frac{\partial\,\text{ppr}_\alpha(\cdot,c)}{\partial \alpha} = \lim_{\alpha\to 0} \frac{\text{ppr}_\alpha(\cdot,c)}{\alpha} = \text{hit}(\cdot,c).$$

[7]A sufficient but not necessary condition for this is that every node except for the "null" node has a small probability of transition to "null". This is a reasonable assumption for our application as the user can always "drop out" with some positive probability.

5.4 Visit

For comparison purposes, we define the visit adfactor which represents the random walk (no restart) visit probability of a node from the "begin" node. This adfactor is not related to the conversion node but simply captures the likelihood of the event happening in a user conversion path. Formally, the adfactor is defined as a row vector $\mathrm{hit}(b,\cdot)$ solving

$$\mathrm{hit}(b,\cdot) = \mathrm{hit}(b,\cdot)\hat{M} + e_b.$$

Similar to Propositon 1, one can show that

$$\lim_{\alpha\to 0} \frac{\partial\,\mathrm{ppr}_\alpha(b,\cdot)}{\partial\alpha} = \mathrm{hit}(b,\cdot).$$

5.5 Marginal Increase (Passthrough)

We introduce the $\mathrm{MI}_\alpha(j)$ adfactor, which represents the *marginal increase in the probability of hitting conversion from the "begin" node b if we increase the weight of all edges outgoing from the node j by ε, allocated in proportion to the current edge weights.* We first compute $\mathrm{MI}_\alpha(j)$ in the context of Personalized PageRank with a restart probability α. Let $\frac{\partial\,\mathrm{ppr}_\alpha(b,c)}{\partial w_{ji}}$ be the marginal influence of an edge weight w_{ji} on the PPR of the "begin" node for the conversion node. The adfactor $\mathrm{MI}_\alpha(j)$ can be proxied by the following summation (note that sensitivity with respect to each outgoing edge is weighted proportionally to the current edge weight).

$$\mathrm{MI}_\alpha(j) = \sum_i w_{ji}\frac{\partial\,\mathrm{ppr}_\alpha(b,c)}{\partial w_{ji}},$$

We first observe that, in the setting of random walks with restart α, $\frac{\partial\,\mathrm{ppr}_\alpha(b,c)}{\partial w_{ij}}$ can be computed by using the following Proposition:

Proposition 2. *The marginal influence of an edge weight on the PPR of the sink c with restart at the source node b is given by a product of the PPR of the start node of the edge (i) with restart at the source node b and the PPR of the sink node c with restart at the end node of the edge (j):*

$$\frac{\partial\,\mathrm{ppr}_\alpha(b,c)}{\partial w_{ij}} = \frac{1-\alpha}{\alpha}\,\mathrm{ppr}_\alpha(b,i)\,\mathrm{ppr}_\alpha(j,c).$$

Using this proposition, we derive a simple closed-form formula for $\mathrm{MI}_\alpha(j)$.

Proposition 3. *The marginal effect of increasing the weight of the outgoing edges of a node j, $MI_\alpha(j)$, is equal to:*

$$MI_\alpha(j) = \frac{\mathrm{ppr}_\alpha(b,j)\,\mathrm{ppr}_\alpha(j,c)}{\alpha}. \qquad (1)$$

The proofs are left to the appendix. An advantage of this closed-form formula is that it lets us apply fast algorithms for computing the MI_α adfactor. The above proposition also implies that, as α tends to zero, the MI adfactor can be computed as follows:

Proposition 4. *The marginal effect of increasing the weight of the outgoing edges from a node j on the hitting probability of conversion, $MI(j)$, is equal to:*

$$MI(j) = \lim_{\alpha\to 0}\frac{MI_\alpha(j)}{\alpha} = \mathrm{hit}(b,j)\,\mathrm{hit}(j,c). \qquad (2)$$

Table 3: Correlation matrix for adfactors. $\alpha = 0.05$

	PPR	Ev.Conv.	LastAd	Visit	Pass
PPR	1.0000	0.9733	0.7285	-0.0657	0.1321
Ev.Conv.	0.9733	1.0000	0.7064	-0.0614	0.1370
LastAd	0.7285	0.7064	1.0000	-0.0542	0.0991
Visit	-0.0657	-0.0614	-0.0542	1.0000	0.8006
Pass	0.1321	0.1370	0.0991	0.8006	1.0000

The above observation implies that the MI adfactor is the same as the *passthrough* adfactor which is *the change in* $\mathrm{hit}(b,c)$ *if we remove the node j from the graph and redirect all incoming edges to this node to the "null" node*, or formally,

$$\mathrm{Pass}(i) = \mathrm{hit}(b,i)\,\mathrm{hit}(i,c).$$

We call the corresponding adfactor the "passthrough" or Pass adfactor, since it captures the value of random walks passing through the evaluated node.

5.6 The Removal Effect

In this part, we introduce the adfactor $\mathrm{RE}(i)$ for each node i which is defined as *the change in the probability of hitting conversion starting from the "begin" node b if we remove node i from the graph.* Intuitively, this adfactor captures the change in the probability of reaching conversion if we remove a node i, or the incoming edges of node i. Similar to the MI_α adfactor, we define the RE_α adfactor in the context of a random walk with restart probability α as

$$\mathrm{RE}_\alpha(i) = \sum_j w_{ji}\frac{\partial\,\mathrm{ppr}_\alpha(b,c)}{\partial w_{ji}},$$

Using Proposition 2, we can derive a similar closed-form formula for $\mathrm{RE}(i)$ which is shown to be equal to $\mathrm{Pass}(i)$.

Proposition 5. *The RE_α and RE adfactors can be computed as follows:*

$$RE_\alpha(i) = \frac{\mathrm{ppr}_\alpha(b,i)\,\mathrm{ppr}_\alpha(i,c)}{\alpha}. \qquad (3)$$

$$RE(i) = \lim_{\alpha\to 0}\frac{RE_\alpha(i)}{\alpha} = \mathrm{hit}(b,i)\,\mathrm{hit}(i,c). \qquad (4)$$

The proofs are left to the appendix. The Pass adfactor is a proxy for both MI and RE adfactors, i.e., $\mathrm{MI}(i) = \mathrm{RE}(i) = \mathrm{Pass}(i)$. As a result, we only report empirical results for the passthrough (Pass) adfactor, and this will imply the same results for the MI and RE adfactors.

5.7 Efficient Algorithms

Computational efficiency is crucial for successful application of any data mining algorithm to the real world advertising data. Fortunately, all adfactors introduced in the previous Section can be efficiently computed on large scale graphs using parallel machines. The key is the observation that the PageRank contribution vectors ($\mathrm{cpr} \equiv \mathrm{ppr}(\cdot,c)$) can be efficiently computed using a local algorithm which adaptively examines only a small portion of the input graph near a specified vertex [4]. We have implemented the algorithm of [4] in the distributed computing environment of MapReduce [11] using the Pregel framework [21]. In the Pregel framework, every node in the graph can perform its local computation and the nodes interact with each other by a periodic exchange of messages. The cpr computation can be achieved by a simple local algorithm (Algorithm 1), in which every node has state consisting of two variables: the currently accumulated cpr and the residual value obtained from other nodes but not yet distributed(resid). The nodes run a simple pushback operation (pushing fraction of the residual to its neighbors over incoming edges) until the value of the residual in each node does not exceed ε. Andersen et al [4] proves the following:

THEOREM 1. *[4] The following algorithm calculates an ε-approximation of the PageRank contribution vector for the conversion node, i.e.* $\mathrm{cpr} \equiv \mathrm{ppr}(\cdot, c)$, *using only* $\frac{1}{\alpha\varepsilon} + 1$ *pushback operations. Moreover, using this algorithm, one can identify the top k nodes with the maximum PageRank contribution using only $O(\frac{k}{\alpha})$ pushback operations.*

Algorithm 1 Local algorithm to calculate $\mathrm{cpr} \equiv \mathrm{ppr}(\cdot, c)$ adfactor of the conversion node c.

```
Initialization:
  $\mathrm{cpr}(u) \Leftarrow 0$, $\mathrm{resid}(u) \Leftarrow 1$ if conversion node otherwise 0
Main Loop:
  while $\exists u$ such that $|\mathrm{resid}(u)| \geq \varepsilon$ do
    Pushback($u, \alpha$)
  end while
Pushback ($u, \alpha$):
  $\mathrm{cpr}(u) \Leftarrow \mathrm{cpr}(u) + \alpha\,\mathrm{resid}(u)$ {accumulate $\alpha$ fraction of the current residual}
  for every incoming edge $w \longrightarrow u$ do
    $\mathrm{resid}(w) \Leftarrow \mathrm{resid}(w) + (1-\alpha)\frac{\mathrm{resid}(u)}{d_{out}(w)}$ {distribute $1-\alpha$ fraction of the current residual to neighbors}
  end for
  $\mathrm{resid}(u) \Leftarrow 0$
```

Using Propositions 3 and 5, one can see that calculation of the MI_α and RE_α, and passthrough (Pass) adfactors requires estimating both the Pagerank contribution vector for conversion (i.e, the $\mathrm{ppr}(\cdot, c)$ value for every node in the graph) and the Personalized PageRank vector of the begin node (the $\mathrm{ppr}(b, \cdot)$ value for every node in the graph). The ε-approximation of the Pagerank contribution vector can be calculated efficiently using pushback operations in Algorithm 1. Moreover, a similar pushforward algorithm developed in [17] can be used for calculation of the ε-approximation of the PPR vector for the begin node. Using Propositions 3, 4, and 5, by multiplying these two vectors, we get the following theorem:

THEOREM 2. *There exists a local ε-approximation algorithm for computing the MI_α and RE_α using $O\left(\frac{1}{\alpha\varepsilon}\right)$ push operations. Moreover, an ε-approximation for Pass, MI, and RE can be computed using $O\left(\frac{1}{\alpha'\varepsilon}\right)$ where $\alpha' = \min_i w_{i,null}$.*

6. EMPIRICAL PROPERTIES OF ADFACTORS

In this section, we present some interesting empirical properties of the adfactors in our dataset. All experiments were done with the "Simple Markov" graph, except where explicitly specified otherwise. The primary reason we chose "Simple Markov" over other adgraph models is that it simplifies interpretation of the results. While similar experiments can be performed with other adgraphs, the results and their interpretation would be different for every particular adgraph model used.

Table 3 presents correlations between different adfactors in our data. As expected, for a sufficiently small value of the restart probability ($\alpha = 0.05$), the PageRank contribution adfactor is highly ($\rho = 0.9773$) correlated to the Eventual Conversion ($\mathrm{hit}(\cdot, c)$) adfactor. There is also a significant but much lower correlation between the PPR factor and the Last Ad adfactor ($\rho = 0.7285$). The visit adfactor is almost uncorrelated to the PPR, eventual conversion and the Last Ad adfactors, indicating that the user queries that occur frequently are not necessarily well connected to the conversion event. Since the passthrough adfactor incorporates the visit adfactor, it is also weakly correlated to the rest of the adfactors. All correlations are statistically significant at 1% level.

Table 4: Stability of the adfactors for the top 1000 nodes.

	Std.Dev.	Mean		Std.Dev.	Mean
PPR	0.0014	0.0123	Visit	0.00005	0.0046
Ev.Conv.	0.0301	0.2796	Pass	0.000007	0.0015
LastAd	0.0272	0.1577			

Table 4 shows stability results (the sample standard deviation) for the adfactors of the top 1,000 graph nodes in a five-fold sample split. One can see that the standard deviations of all adfactors are relatively low compared to the mean values suggesting that the adfactors of the most important nodes in the graph are relatively stable in the presence of sampling variance even though the edge weights in the graph can be volatile (Table 1).

In the rest of this Section, we evaluate how adfactors correlate with other node attributes such as brand/non-brand [8], click/impression and broad/exact match. Note that in all three cases the evaluated attribute is a binary variable, thus, instead of presenting simple correlations, one can deliver better intuition by using ROC curves for the corresponding classifier. Consider, for instance, Figure 3. For every adfactor, one can construct a classifier of node "brand-iness" by using the adfactor $< T$ decision rule: if the adfactor is larger than the threshold T, classify it as brand (or non-brand depending on which one is better), otherwise classify it as nonbrand (brand). Varying values of T, one can achieve different (precision, recall) points; all such points constitute the ROC curve.

Figure 3 shows that the PPR adfactor is an excellent predictor of "brand-iness" for impressions (AUC = 0.95509), even stronger than the last ad score (AUC = 0.89930). At the same time, for clicks the conversion Pass adfactor gives the best predictor of the brand (AUC = 0.87707), although it is an extremely weak predictor for impressions (AUC = 0.57120). We suggest that users who search with brand queries are (naturally) much more likely to convert with the advertiser, than users that search with generic queries, thus the PPR adfactor measuring structural correlation to the conversion event is a good signal of the brand nature of the user query. Yet, once the user clicks on the advertiser's ad, the signal loses its value, because, once the ad attracted enough user attention, the query that triggered the ad in the first place becomes much less important. The good predictive power of the Pass adfactor on click but not impression "brand-iness" suggests that users are more likely to convert with the advertiser through click on a brand query ad rather than through click on a non-brand query ad, however converting users get equal exposure to both brand and non-brand impressions before they convert.

[8]Queries were marked as *brand* if they included a phrase with the edit distance of at most three to the advertiser's name or the brand name, and marked as *non-brand* otherwise. The results were manually inspected to ensure that the mapping is reasonable.

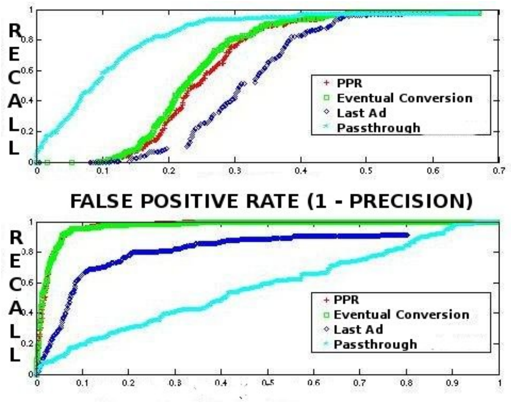

Figure 3: ROC curve for brand prediction for clicks (top) and impressions (bottom)

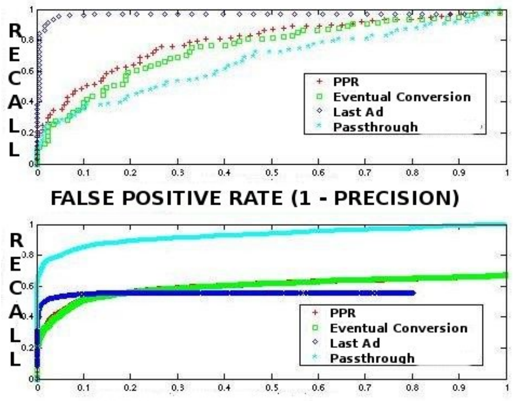

Figure 4: ROC curve for click prediction for brand queries (top) and nonbrand queries (bottom)

Figure 4 shows that for brand queries the Last Ad adfactor and the PPR adfactor are excellent signals of the click attribute; for non-brand queries they predict well for recall of up to approximately 0.6, after which the precision-recall curve becomes flat (until the eventual jump to (1.0,1.0)). This weird behavior can be attributed to the fact that, in our dataset, we observe a number of rare generic (non-brand specific) queries, for which click on the ad does not lead to any conversion. We emphasize that such queries are rare, therefore using the Pass adfactor filters them out and the Pass adfactor shows excellent predictive power for the non-brand queries in Figure 4.

Next, we compare the PPR and the Last Ad adfactors by using a difference between the ad rank in both models as a predictor for brand/non-brand, click/impression and

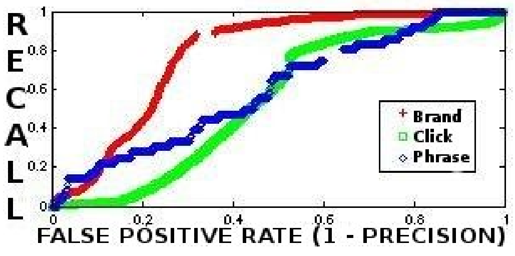

Figure 5: ROC curve for (PPR rank - last ad rank) predictor

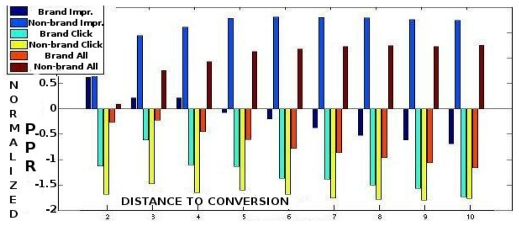

Figure 6: PPR behavior as a function of distance to conversion

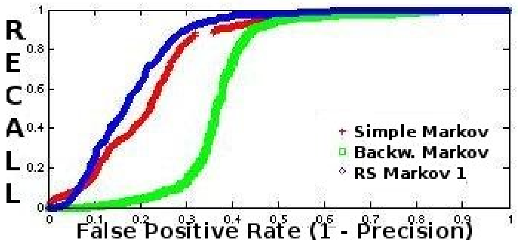

Figure 7: ROC curve for brand prediction by different models

broad/exact attributes. Results in Figure 5 suggest that the difference is a good predictor of the brand attribute; manual inspection shows that this is because the PPR adfactor ranks brand nodes even higher than the Last Ad adfactor. On the other hand, the ROC curves for click and phrase(broad)/exact match prediction are close to diagonal suggesting that there is no systematic difference in rankings of clicks/impressions and broad/exact match nodes by both methods.

Next, we plot the PPR behavior as a function of the distance to conversion in Figure 6. The figure was constructed using the data from the "Backward Markov" model, in which every user query is represented by multiple nodes in the graph, depending on its distance from the conversion node. Thus, for every query q, one can calculate

$$\log(\text{ppr}(q \text{ at distance } d, c)) - \log(\text{ppr}(q \text{ at distance } 1, c))\ [9].$$

Figure 6 shows average of these results for different groups. Note that for clicks the PPR values at distance two and more from conversion are significantly smaller than the PPR values at distance one, suggesting that if the user clicks on the ad and does not convert immediately, the likelihood to convert in the future goes down. Consistent with our prior observations, the PPR behavior for brand and non-brand clicks is similar. On the other hand, for non-brand impressions the PPR grows (or at least doesn't decrease) with distance to conversion, indicating that exposure to the advertiser ad on non-brand queries can have a long-term positive correlation with the likelihood of the user to convert. For brand impressions, the PPR decreases with distance to conversion, although not as strongly as for clicks, again suggesting that, if the user searches for a brand and does not convert soon enough, the likelihood of converting in the future goes down.

Finally, Figure 7 investigates how different graph structures affect the relationship between the PPR adfactor and the brand attribute of the node. We consider three models: "Simple Markov", "Backward Markov" and "RS Markov 1".

[9]The log representation was primarily chosen to reduce sensitivity to outliers.

For "Simple Markov" model we simply plot the ROC curve using the PPR adfactor as a predictor for brand. For "Backward Markov" model we use the observation (from Figure 6) that the PPR of brand nodes decays with distance from conversion while the PPR of nonbrand nodes grows with distance from conversion, thus we use

$$\sum_{d=2}^{10} (\log(\mathrm{ppr}(q \text{ at distance } d, c)) - \log(\mathrm{ppr}(q \text{ at distance } 1, c)))$$

as the predictor. For the "RS Markov 1" model, we would expect similar behavior to hold and the PPR of the brand nodes in the "search" state (before the user clicked on any ads) to be lower than that in the "interested" state (after the user clicked on some ad), while the PPR of the nonbrand nodes in the "interested" state to be higher than that is the "search" state; thus, we construct the predictor as

$$\log(\mathrm{ppr}(q \text{ in search state}, c)) - \log(\mathrm{ppr}(q \text{ in interested state}, c))\,.$$

Figure 7 shows that the "RS Markov 1" model has the best predictive power for the brand attribute, confirming the intuition that user clicks have different value in brand and non-brand contexts.

7. DATA MINING TOOLS

Producing intuitive and concise reports from huge amounts of data is the desired goal for advertisers. In this section, we discuss various data mining tools that can be developed using our ad factors defined on the graph models introduced in the paper.

Identifying the top k ads with the largest adfactor. The simplest type of report one can think of is identifying the top k ads with the largest adfactors in a particular graph model. The value of such report depends on the particular adfactor and the underlying graph model used. For instance, in Section 6, we show that the PPR adfactor (in the "Simple Markov" graph) has high but not perfect correlation with the brand attribute of the ad impression. Identifying the top k impressions with the PPR adfactor can thus be interpreted as identifying the top k impressions (user queries) with the highest "branding" impact: many of these user queries will in fact have the advertiser name or the brand name in them, but some will not and thus can be thought of as "shadow brands". Due to privacy issues, we cannot show an actual example of such output. Another interesting example of the top k output is the top k impressions for the Pass adfactor; they can be thought of as the top k user queries associated with the largest revenue to the advertiser if one takes into account the long-term correlations in the graph. Interestingly, from Figure 3, we know that these are uncorrelated with the brand attribute.

Identifying ads with significant long-term effects not taken into account by the Last Ad model. Advertisers traditionally rely on the Last Ad reports (clickthrough and conversion rates) to evaluate ad effectiveness. An alternative report we suggest would be to look for ads that are ranked higher with adfactors taking into account the long-term correlations (like PPR) than with the Last Ad adfactor. One such criteria for ranking is the difference between the PPR adfactor node rank and the Last Ad adfactor node rank (properties of this criteria are shown in Figure 5). Another criteria is to look for impressions that either have the PPR growing with the distance from conversion ("Backward Markov" model) or have the PPR in the "search" state higher than in the "interested" state ("RS Markov 1" model). As Figure 7 shows, the second approach is more biased towards the brand impressions than the first one.

Identifying the top k ads with the maximum marginal increase (MI) adfactor. A popular advertising objective is to maximize the expected number of user conversions given a certain budget constraint. In a random walk model, we can restate it as maximizing the probability of hitting the conversion node starting from the "begin" node. A heuristic way to identify the valuable nodes to invest on is to examine a small increase on the bid of a query (or keyword) which will have the maximum effect on the probability of hitting the conversion node. This small increase on the bid results in a small increase in the weight of incoming edges for this query. Thus, in order to identify queries for which the increase in their bid results in the maximum marginal increase in the probability of hitting conversion, we can look for the top k ads with the maximum MI adfactor, and the advertiser may consider increasing the bid for queries with large MI. [10]

Explaining the adfactor value The adfactor assigned to any particular node in the graph, must be easily explained by a local structure around this node. For instance, for the PPR (PageRank contribution) adfactor of a node u, one can always consider the top-m neighbor nodes through which the explained node u contributes the largest fraction of the PageRank to the conversion node. Formally, let π_u be the permutation of all neighbors of the node u that arranges them in the decreasing order of $w_{uv}\,\mathrm{ppr}_\alpha(v, c)$. We define $\pi_u(m)$ as the set of the first m nodes in the permutation. These nodes can be thought of as the most likely next actions of the user after the explained action, assuming that the user is going to convert. It is straightforward to extend the pushback Algorithm 1 to keep track of the top m contributors for every node, thus one can efficiently generate corresponding reports. The reports are best represented as graph plots, constructed starting at a seed node s and going up to a distance d in a graph of the top-m neighbors of every node. Due to privacy reasons as well as the space limit, we omit an example of such plot.

8. CONCLUSIONS

Online advertisers have access to aggregate statistics such as the clickthrough rate, the conversion rate or the lift of their ad campaigns, but seek to understand more sophisticated correlations in users' trajectories of ads seen and users' actions. Here we introduce an alternative data mining technique for aggregating user-level advertising data. We define various adgraphs to model the data pertinent to the advertiser and propose several adfactors based on stationary probabilities of suitable random walks to quantify the impact of ad events. We show anecdotal evidence for adfactors and describe their potential data mining applications.

Our research introduces novel primitives for mining advertiser-specific data for ad effects. Adfactors we define are succinct, easy to compute and capture many structural properties of

[10]We emphasize that this is only a heuristic. Any practical bidding strategy should also take into account the current bidding landscape (how much do the advertiser and the competitors bid) for a particular keyword.

users' trajectories. In the future, richer data mining primitives can be developed using adfactors, which may potentially reveal more sophisticated correlations in user behavior, including negative correlations. Other valuable directions for future research include developing richer graphical models of user behavior, beyond the Markov models in this paper, and adapting adfactors to them.

Acknowledgements. We would like to thank Chao Cai, Zhimin He, and Sissie Hsiao for helping us with the data sets, and Sheng Ma, Fernando Pereira, R. Srikant for interesting discussions.

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, 2006.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, pages 487–499, 1994.

[3] H. Akaike. A new look at the statistical model identi↓cation. *IEEE Transactions on Automatic Control*, 19:716– 723, Dec 1974.

[4] R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, V. Mirrokni, and S.-H. Teng. Local computation of pagerank contributions. *Internet Mathematics*, 5:23–45, Jan. 2009.

[5] Atlas. Engagement mapping. Thought Paper, Available at `http://www.atlassolutions.com/uploadedFiles/Atlas/Atlas\_Institute/Engagement\_Mapping/eMapping-TP.pdf`, 2008.

[6] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th International Conference on World Wide Web, WWW*, pages 895–904, 2008.

[7] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 191–200, 2009.

[8] comScore. Whither the click? comscore brand metrix norms prove 'view-thru' value of on-line advertising. Available at http://www.comscore.com/press/release.asp?press=2587, 2008.

[9] G. Cormode, F. Korn, S. Muthukrishnan, and Y. Wu. On signatures for communication graphs. In *Proceedings of the 24th International Conference on Data Engineering, ICDE*, pages 189–198, 2008.

[10] E. E. Dar, Y. Mansour, V. Mirrokni, S. Muthukrishnan, and U. Nadav. Budget optimization for broad-match ad auctions. In *WWW, World Wide Web Conference*, 2009.

[11] J. Dean and S. Ghemawat. Mapreduce: simpli↓ed data processing on large clusters. *Communications of ACM*, 51(1):107–113, 2008.

[12] T. Fawcett and F. J. Provost. Combining data mining and machine learning for e↑ective user pro↓ling. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 8–13, 1996.

[13] J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein. Budget optimization in search-based advertising auctions. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 40–49, 2007.

[14] A. Hassan, R. Jones, and K. Klinkner. Beyond DCG: User behavior as a predictor of a successful search. In *WSDM '10: Proceedings of the 3rd international conference on web search and data mining*, 2010. Forthcoming.

[15] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

[16] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000.

[17] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003.

[18] K. Keller. Brand equity and integrated communication. In *Integrated Communication: Synergy of Persuasive Voices*, 1996.

[19] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.

[20] R. Lewis and D. Reiley. Retail advertising works!: Measuring the e↑ects of advertising on sales via a controlled experiment on yahoo! Working paper, Yahoo! Research, 2009.

[21] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *PODC '09: Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 6–6, 2009.

[22] H. Mannila, H. Toivonen, and I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, Sept. 1997.

[23] S. Muthukrishnan, M. Pal, and Z. Svitkina. Stochastic models for budget optimization in search-based advertising. In *Lecture Notes in Computer Science, Internet and Network Economics*, pages 131–142, 2007.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report. Stanford InfoLab., 1999.

[25] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user pro↓le constructed without any e↑ort from users. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 675–684, 2004.

10. APPENDIX

Proof of Proposition 1

PROOF. From [4]

$$\mathrm{ppr}_\alpha(\cdot, v) = \alpha e_v \sum_{t=0}^{\infty} (1-\alpha)^t (M^T)^t.$$

After taking the derivative

$$\frac{\partial\, \mathrm{ppr}_\alpha(\cdot, c)}{\partial \alpha} = \frac{\mathrm{ppr}_\alpha(\cdot, c)}{\alpha} + \alpha e_c \sum_{t=1}^{\infty} t(1-\alpha)^{t-1} (M^T)^t.$$

It immediately follows that

$$\left\| \frac{\partial\, \mathrm{ppr}_\alpha(\cdot, c)}{\partial \alpha} - \frac{\mathrm{ppr}_\alpha(\cdot, c)}{\alpha} \right\|_\infty \to 0.$$

Next, we have $h = e_c + hM^T$ and

$$\mathrm{ppr}_\alpha(\cdot, c) = \alpha e_c + (1-\alpha)\,\mathrm{ppr}_\alpha(\cdot, c) M^T,$$

which is equivalent to

$$\frac{\mathrm{ppr}_\alpha(\cdot, c)}{\alpha} = e_c + \frac{1-\alpha}{\alpha}\,\mathrm{ppr}_\alpha(\cdot, c) M^T.$$

After substraction

$$h - \frac{\mathrm{ppr}_\alpha(\cdot, c)}{\alpha} = \left(h - \frac{1-\alpha}{\alpha}\,\mathrm{ppr}_\alpha(\cdot, c)\right) M^T,$$

or

$$\left(h - \frac{\mathrm{ppr}_\alpha(\cdot, c)}{\alpha}\right) = \mathrm{ppr}_\alpha(\cdot, c) M^T \left(I - M^T\right)^{-1} \to 0,$$

as $\mathrm{ppr}_\alpha(\cdot, c) \to 0$. □

Proof of Proposition 2

PROOF. We know that the ppr vector p with restart at node b solves the equation $p = \alpha e_b + (1-\alpha)pM$, where M is the transition matrix. Fix nodes i and j and take a derivative with respect to w_{ij}:

$$\frac{\partial p}{\partial w_{ij}} = (1-\alpha)p\frac{\partial M}{\partial w_{ij}} + (1-\alpha)\frac{\partial p}{\partial w_{ij}} M.$$

Now, $\frac{\partial M}{\partial w_{ij}}$ is a zero matrix with a single 1 at row i and column j. It follows that

$$\frac{\partial p}{\partial w_{ij}} = (1-\alpha)p[i]e_j + (1-\alpha)\frac{\partial p}{\partial w_{ij}} M,$$

where $p[i]$ is the i-th component of the vector p. This can be written as

$$\frac{\partial p}{\partial w_{ij}} = \left(\frac{1-\alpha}{\alpha} p[i]\right) \alpha e_j + (1-\alpha)\frac{\partial p}{\partial w_{ij}} M,$$

and by linearity of the PPR we can see that this is just $\left(\frac{1-\alpha}{\alpha} p[i]\right)$ times the PPR vector with restart at node j, i.e.,

$$\frac{\partial\, \mathrm{ppr}_\alpha(b, c)}{\partial w_{ij}} = \frac{1-\alpha}{\alpha}\,\mathrm{ppr}_\alpha(b, i)\,\mathrm{ppr}_\alpha(j, c).$$

□

Proofs of Propositions 3 and 5

PROOF. For outgoing edges, we can use the equality $\mathrm{ppr}_\alpha(i, c) = \alpha I(i == c) + (1-\alpha)\,\mathrm{ppr}_\alpha(j, c) w_{ij}$ [4], as shown below:

$$\begin{aligned}
\sum_j w_{ij} \frac{\partial\, \mathrm{ppr}_\alpha(b, c)}{\partial w_{ij}} &= \frac{(1-\alpha)\,\mathrm{ppr}_\alpha(b, i)}{\alpha} \left(\sum_j w_{ij}\,\mathrm{ppr}_\alpha(j, c)\right) \\
&= \frac{1-\alpha}{\alpha}\,\mathrm{ppr}_\alpha(b, i) \frac{1}{1-\alpha}\,\mathrm{ppr}_\alpha(i, c) \\
&= \frac{\mathrm{ppr}_\alpha(b, i)\,\mathrm{ppr}_\alpha(i, c)}{\alpha}.
\end{aligned}$$

For incoming edges, we can use similar trick with $\mathrm{ppr}_\alpha(b, i) = \alpha I(i == b) + (1-\alpha)\,\mathrm{ppr}_\alpha(b, j) w_{ji}$. □

Proof of Theorem 2

PROOF. MI_α and RE_α can be approximated by Theorem 1 of [4]. For MI, RE and Pass simply note that $\hat{M}$ can be written as $(1-\alpha')M^*$, where M^* is a valid random walk matrix (every row sums to at most 1.0, the rest goes to "null"). One can therefore calculate hit and all derivative adfactors (MI, RE and Pass) via a pushback algorithm for the PPR of a random walk with the restart probability α' so Theorem 1 of [4] works again. □