# A Search-based Method for Forecasting Ad Impression in Contextual Advertising

Xuerui Wang
University of Massachusetts
Amherst, MA
xuerui@cs.umass.edu

Andrei Broder
Yahoo! Research
Santa Clara, CA
broder@yahoo-inc.com

Marcus Fontoura
Yahoo! Research
Santa Clara, CA
marcusf@yahoo-inc.com

Vanja Josifovski
Yahoo! Research
Santa Clara, CA
vanjaj@yahoo-inc.com

## ABSTRACT

*Contextual advertising* (also called *content match*) refers to the placement of small textual ads within the content of a generic web page. It has become a significant source of revenue for publishers ranging from individual bloggers to major newspapers. At the same time it is an important way for advertisers to reach their intended audience. This reach depends on the total number of exposures of the ad (*impressions*) and its *click-through-rate* (CTR) that can be viewed as the probability of an end-user clicking on the ad when shown. These two orthogonal, critical factors are both difficult to estimate and even individually can still be very informative and useful in planning and budgeting advertising campaigns.

In this paper, we address the problem of forecasting the number of impressions for new or changed ads in the system. Producing such forecasts, even within large margins of error, is quite challenging: 1) ad selection in contextual advertising is a complicated process based on tens or even hundreds of page and ad features; 2) the publishers' content and traffic vary over time; and 3) the scale of the problem is daunting: over a course of a week it involves billions of impressions, hundreds of millions of distinct pages, hundreds of millions of ads, and varying bids of other competing advertisers. We tackle these complexities by simulating the presence of a given ad with its associated bid over weeks of historical data. We obtain an impression estimate by counting how many times the ad would have been displayed if it were in the system over that period of time. We estimate this count by an efficient two-level search algorithm over the distinct pages in the data set. Experimental results show that our approach can accurately forecast the expected number of impressions of contextual ads in real time. We also show how this method can be used in tools for bid selection and ad evaluation.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval


*WWW 2009*, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

## General Terms

Algorithms, Experimentation, Measurement, Performance

## Keywords

Online advertising, contextual advertising, content match, impression forecasting, WAND

## 1. INTRODUCTION

Web advertising has become a major industry: according to a recent study by the Interactive Advertising Bureau and PricewaterhouseCoopers International, online advertising spending in 2007 reached over 21 billion US dollars, an increase of 26% from the previous year [12]. This is the first time that online advertising spending was larger than each of the traditional advertising media: radio, cable and broadcast TV. A significant part of this market consists of *textual ads*, the ubiquitous short text messages usually marked as "sponsored links" or similar. The main advertising channels used to distribute textual ads are:

1. *Sponsored Search* (SS) or *Paid Search Advertising* which consists in placing ads on the result pages from a web search engine, with ads driven by the originating query. All major current web search engines (Google, Yahoo!, Microsoft, etc.) support such ads and act simultaneously as a search engine and an ad agency.

2. *Contextual Advertising* (CA) or *Content Match* which refers to the placement of commercial ads within the content of a generic web page. In contextual advertising, usually there is a commercial intermediary, called an *ad-network*, in charge of optimizing the ad selection with the twin goal of increasing revenue (shared between publisher and ad-network) and improving user experience. Again, all major current web search engines provide such ad-networking services but there are also many smaller players.

The prevalent pricing model for textual ads is that the advertisers pay a certain amount for every click on the advertisement (pay-per-click or PPC). There are also other models: pay-per-impression where the advertisers pay for the number of exposures of an ad, and pay-per-action where the advertisers pay only if the ad leads to a sale or similar transaction. For simplicity, in this paper, we consider only

the PPC model. In this model, the total amount paid by an advertiser (and hence the revenue of the search engine or ad-network) is related to the total number of exposures of the ad (impressions) and the *click-through-rate* (CTR) which can be viewed as the probability that an end-user would click on the ad when shown. Both impressions and CTR are difficult to estimate due to various factors such as traffic fluctuation and sparsity, but they are orthogonal. An ad with a high CTR might not be shown enough to generate revenue, and on the other hand, ads with low CTRs could get tons of impressions and waste expensive traffic. However any of them individually can still be very informative and extremely useful in planning and budgeting advertising campaigns. We focus on impression forecasting in this paper.

In the SS market, textual ads are characterized by *bid phrases* representing those queries where the advertisers would like to have their ads displayed. The amount paid by the advertiser for each ad click is determined by an auction process [7]. Very simplified, in this process each advertiser enters a bid for the phrase of interest; the search engines choose among the ads competing on that phrase those ads that have high bids and high CTRs, thus attempting to maximize the search engines' total revenue. Of course CTR is not known *a priori*; however it can be determined, e.g., by historical observation, or approximately estimated .

The situation is more complicated in contextual advertising. Here the bid phrase has no direct bearing on the ad placement: the ad network has to find ads that will have high CTRs for the given context. Given a page, rather than placing generic ads, it seems preferable to have ads related to the content to match the user interest and thus to increase the probability of clicks. This intuition is supported by the analogy to conventional publishing where there are very successful magazines (e.g., *Vogue*) where a majority of the content is topical advertising (fashion in the case of *Vogue*) and by user studies that have confirmed that higher relevance increases the number of clicks [6, 18].

The majority of the approaches proposed so far for estimating the relevance of a given ad to a given content, and thus indirectly CTR, are based on the co-occurrence of words or phrases within ads and pages [13, 16, 20] or on a combination of semantic and syntactic factors [4].

At the core, most of these approaches can be viewed as computing a similarity score $\text{Sim}_{a,p}$ between a vector of features characterizing the ad $a$ and a vector of features characterizing the page $p$. For the ad $a$ such features could include the bid phrase, the title words (usually displayed in a bold font in the presentation), synonyms of these words, the displayed abstract, the target URL, the target web site, the semantic category, etc. Similarly, for the page $p$ such features might include words on page, words in title, URL, category, synonyms, etc. In this paper, for simplicity, we assume that given a page $p$ and a set of available ads $\{a_1, a_2, \cdots\}$, the ad network places the ads that maximize the product

$$\text{Score}_{a,p} \stackrel{\text{def}}{=} \text{Sim}_{a,p} \times \text{Bid}_a, \tag{1}$$

where $\text{Bid}_a$ is the bid amount of ad $a$ in the PPC model. As we discussed, note that $\text{Sim}_{a,p}$ can be regarded roughly as an unnormalized approximation of CTR [6, 18]. In other words, ad placement discussed here indeed takes CTR *indirectly* into consideration.

Given the model above, advertisers eager to reach as many customers as possible, have two possibilities: they can either craft their ads to better match more opportunities or they can increase their bids. In either case, the advertisers are probably interested to know what is the effect of these changes. A simple approach is to try a test ad and/or a test bid in real traffic and analyze the effect after a number of days. This is of course very expensive, inefficient, and has a very long turn-around and a huge variance. The need for forecasting is actually well recognized: existing SS systems provide such facilities,[1] but none of the available CA systems provides forecasting, to the best of our knowledge. Such a forecasting system has obvious applications for ad selection, campaign budgeting, and advertising strategy.

The main goal of this paper is to propose a method to forecast the future performance of new and modified ads accurately, in a real-time manner, based on replaying past impression data. The past data is composed of actual page views with the associated ads (there are multiple ads shown on the page). To estimate the impression count, given an ad $a$ and a bid $\text{Bid}_a$, we consider all content match opportunities over, say, the last four weeks, and check how often the ad $a$ would have been shown if it were in the system with the given bid. That is, how often $\text{Score}_{a,p}$ would have exceeded the score of the lowest scoring ad actually shown on the page $p$. The impression rate is then used in an obvious manner to predict the impression volume over, say, the next week.

At first blush this task seems daunting: for a real system we would need to check billions of past CA opportunities. To address this problem the efficiency of our algorithm depends only on the number of distinct pages, which is a relatively small number when compared to the total number of impresssions. Secondly, our search procedure greatly reduces the number of pages that need to be considered for full evaluation.

We verified our approach using a set of 700M actual CA opportunities over around 375K distinct pages obtained from a certain random set of hosts participating in Yahoo! ad network. The ad database used for our experiments consisted of about 10M actual ads and their bids. The ranking function is given by Formula (1), and the similarity score by a formula related to the approach presented in [4]. With this data, and using a standard PC with 2GB memory, the time required to forecast the volume of impressions for a fixed ad and a single bid is around 10 milliseconds; plotting a curve forecasting the ad volume for 100 different bid levels takes less than 400 milliseconds. While it can be argued that a real system needs to be hundreds of times larger, our system is trivially parallelizable, and its performance depends only on the number of distinct pages in the historical data. Furthermore, as long as the page index fits in memory, the forecasting time increases very slowly—our index is less than 10MB and thus we estimate that a single 8GB box can handle 25 million distinct pages in 15 milliseconds. Last but not least, infrequent pages can be ignored.

In this paper we focus on ad selection based on similarity score. Real-world matching systems involve many ad selection mechanisms such as business rules, behavioral targeting (different users might get different ads), advertiser budgets, pricing based on a generalized second price auction, traffic quality discounts, positional effects (same ad exhibits different CTRs in different positions), anti-fraud measures, and so on. Some of these could be easily included in our model

[1] e.g., http://searchmarketing.yahoo.com/srch/features.php

(e.g., behavioral targeting), and some can be captured in the training data (e.g., anti-fraud measures). The approach of replaying ads can in principle accomodate any ad selection technique as long as we can efficiently invert the selection process from selecting ads for a given page, to counting the pages where the ad would have been shown if it were present in the system.

## 2. OVERVIEW OF OUR APPROACH

Given a new ad $a$ and a bid $\text{Bid}_a$, we consider all impressions over some training period, and check how often the ad $a$ would have been shown if it were present in the system with the bid. In other words we count how often $\text{Score}_{a,p}$ exceeds the score of the lowest scoring ad that the system would assign to page $p$ absent ad $a$. This minimum score is denoted as $\text{minScore}_p$ and is discussed further below.

The abstract problem that we need to solve is as follows: given a feature vector $u$, and a stream of events $\{X_i\}$ each having an associated feature vector $v_i$ and a threshold $t_i$, we want to compute $\sum_{(u,v_i)>t_i} 1$. In our case, $u$ represents the features associated with the test ad $a$, $v_i$ represents the features of the page $p_i$, the threshold $t_i$ is $\text{minScore}_{p_i}$, and $(u, v_i)$ is $\text{Score}_{a,p_i}$.

In contextual advertising, the number of ads shown on a page typically varies between 1 and 20. For a page $p_i$, the ad ranking system chooses the top $k_i$ ranking ads according to Formula (1). The score of the lowest ranking ad among these $k_i$ ads shown on $p_i$ defines $\text{minScore}_{p_i}$. If the test ad $a$ would have been available when $p_i$ was displayed, then $a$ would have been shown if and only if $\text{Score}_{a,p_i} > \text{minScore}_{p_i}$.[2] Naively, we can go over every page in the training set and we can calculate how many times the test ad would have been shown. Since the ad database might include hundreds of millions of ads, and the training set possibly include billions of impressions, linear scanning is completely impractical.

To reduce the size of the problem, we assume that $\text{Score}_{a,p}$ is constant over all occurrences of the page $p$. However $\text{minScore}_p$ might vary from occurrence to occurrence because ads are continuously added and deleted from the system and bids change as well. Assume that the page was shown $n$ times. There are several possible approaches to circumventing the multiple minima problem:

1. We keep, for each page $p$, the distribution of minimum scores, that is, we keep a list of minima $m_1 \le m_2 \le \cdots \le m_r$ and of weights $0 < w_1 \le w_2 \le \cdots \le w_r = \text{Imp}_p$, where $\text{Imp}_p$ is the number of impressions of page $p$ . The interpretation is that there were a total of $w_1$ occurrences of $p$ with $\text{minScore}_p = m_1$, there were a total of $w_2$ occurrences of $p$ with $\text{minScore}_p = m_1$ or $m_2$, etc. Thus if $\text{Score}_{a,p}$ falls between $m_j$ and $m_{j+1}$, the ad $a$ would have been shown $w_j$ times.

2. We set $\text{minScore}_p$ to the median value of the minimum ad scores for the page impressions.

3. We ignore the historical $\text{minScore}_p$ and recompute it considering only the currently available ads and their bids. This has the advantage that it captures the most recent picture of the marketplace but it does not reflect the historical bid variation.

[2]When $\text{Score}_{a,p_i} = \text{minScore}_{p_i}$, which is extremely rare, we assume the test ad would not have been shown.

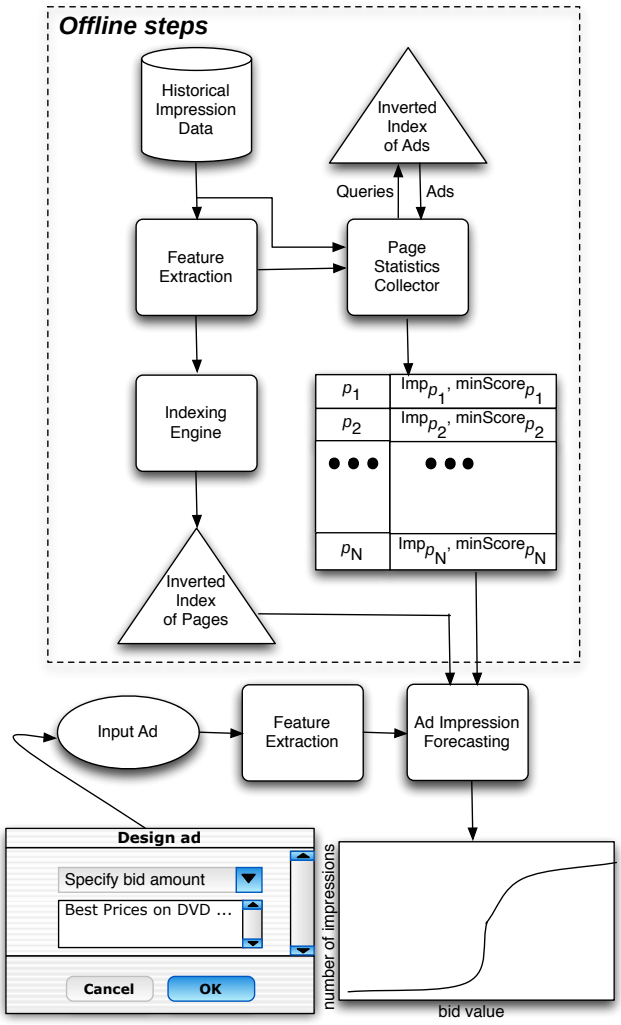


**Figure 1: The system architecture for forecasting ad impressions. The part inside the dotted line runs offline and prepares the data for the online forecasting. The output can be the number of forecasted impressions or an impression vs. bid curve.**

Preliminary experiments show that there is no significant difference between different approaches. In our experiments, we used the last approach, but the algorithms can be easily modified to use approach 1 or 2.

By collapsing all occurrences of one page to one record we achieve a huge improvement (by a factor of about 2000 in our experiments); nevertheless, even a linear scan of all distinct pages is prohibitively time consuming. For further improvement, we reduce this scanning process to a two-level search process over the collection of distinct web pages.

At the first level, an inexpensive approximate evaluation is conducted to identify the set of pages on which the ad could have been shown. The pages in this set are called *candidates*. At the second level, a full evaluation is performed for each candidate page by comparing the score of the ad being considered against $\text{minScore}_p$ of the candidate $p$. If the score is larger, then the forecast counter is increased accordingly.

Our system architecture is presented in Figure 1. The upper part of the picture depicts offline processing that consists

of analyzing the pages, building a page inverted index, and creating a page statistics file. The page analysis module extracts, for each page $p$, the features needed to compute $\text{Sim}_{a,p}$. Since we are using the approach 3 above, these features are used to obtain the top $k$ ads from the ads currently active in the system and thus compute $\text{minScore}_p$. The inverted index is a mapping from features to the pages that contain these features (a description of inverted indexes is given in Section 3.1). The page statistics file contains, for each page $p$, the number of its impressions in the training set and $\text{minScore}_p$.

In the online component of the system we use the inverted page index and the page statistics file to forecast the number of impressions of a given ad. When an advertiser submits a new ad, we extract features from this ad and use these features to query the page index via the two-level search process described above. Advertisers can either provide a bid value for the ad being tested and get back the number of expected impressions for this specific ad/bid combination, or they can ask for an impression vs. bid curve, as illustrated in Figure 1. An advertiser can also use this interface to change the wording of its ad and then check the expected number of impressions with a different ad description.

# 3. FORECASTING AD IMPRESSIONS AS A SEARCH PROCESS

To reduce the cost of the search over the past impressions we use a document-at-a-time (DAAT) strategy [17] and a two-level evaluation approach. Here, we first go over all the web pages and identify candidates by inexpensive approximate evaluation. Full evaluation is performed only over the candidates by checking if the given test ad would have been shown if it were present in the ad database. The check calculates the score that the test ad would have had if it were present in the ad database at the time of the page impression. Then the score is compared to the minimal score of the ads actually shown on the candidate page $\text{minScore}_p$. We have adapted the WAND [3] procedure to avoid examining every page in the data set.

For reference, Table 1 summarizes all the notation used in this paper.

## 3.1 Background

Here we briefly review some basic concepts in information retrieval (IR) and terminology used in this paper.

***Inverted Index.*** Most information retrieval (IR) systems use inverted indexes as the main data structure for full-text indexing [19]. There is a considerable body of literature on efficient ways of building inverted indexes (e.g., [1, 2, 9, 11, 15, 19]) and using them to evaluate full-text queries (e.g., [3, 14, 19]).

In this paper, we use an inverted index structure where each occurrence of a feature $f$ within a web page $p$ is represented by a *posting* of a form $\langle \text{PID}, w_{f,p} \rangle$. PID is the page identifier which in this paper we assume that ranges from 1 to the number of unique pages $N$. $w_{f,p}$ is called *weight* and is used to store arbitrary information about the occurrence of $f$ within $p$. The postings associated with the same feature are grouped into a *posting list*. Each posting list is sorted in increasing order of PID. Often, skip lists or B-trees [10] are used to index the posting lists [9, 15] to allow for quick finding of postings given a PID.

**Table 1: Notation used in this paper**

| SYMBOL | DESCRIPTION |
|---|---|
| $N$ | the total number of distinct web pages |
| $p_i$ | the $i^{th}$ web page |
| $f_j$ | the $j^{th}$ feature, such as a unigram term |
| $w_{f,p}$ | the weight of feature $f$ in page or ad $p$ |
| $\text{Imp}_p$ | the number of impressions of web page $p$ |
| $\text{Sim}_{a,p}$ | the similarity between ad $a$ and page $p$ |
| $\text{Bid}_a$ | the bid amount of ad $a$ |
| $\text{Score}_{a,p}$ | the ranking score for ad $a$ on page $p$ |
| $\text{maxWeight}_f$ | the largest weight of feature $f$ in any page |
| $\text{minScore}_p$ | the lowest ranking score of previously shown ads on page $p$ |

***Joining Posting Lists*** Typically, to evaluate the query, we need to perform a join over the pre-sorted posting lists. One of the common approaches to performing this join is to use a document-at-the-time (DAAT) evaluation. Most of the DAAT algorithms in the literature are merge-joins with improved efficiency [10]. To evaluate a free-text query using a DAAT algorithm, a *cursor* $C_f$ is created for each feature $f$ in the query, and is used to access $f$'s posting list. During the join, the cursors are moved in a coordinated fashion forward over the posting lists. Almost all posting list join algorithms are based on the $C_f.\textit{beyond}(\text{PID } p)$ primitive that advances the $C_f$ cursor to the first posting such that PID is greater than or equal to $p$, returning the PID of the new position. If $p$ is beyond the end of the posting list, *beyond*() returns a special posting element with an identifier *LastID* which is the smallest integer $(N+1)$ larger than all existing PIDs in the index.

***Scoring.*** The join algorithm evaluates the query over a subset of candidate pages by calculating the scores between the query and the pages. The match between the query and the page is mostly quantified by a query dependent or dynamic score. Usually, there is also a query-independent component which is based on the *static rank* of the web page. In most IR systems, the query-dependent component of the score follows an additive scoring model for each feature, whereas the static component can be based on the connectivity of web pages, as in PageRank [2], or on other factors such as source, length, creation date, etc. In this work, we do not use static scores for pages. The similarity score between page $p$ and ad $a$, $\text{Sim}_{a,p}$ is decided by summing contributions over all common features of $a$ and $p$ as follows:

$$\text{Sim}_{a,p} \stackrel{\text{def}}{=} \sum_{f \in a \cap p} w_{f,a} \times w_{f,p}, \tag{2}$$

where the weights $w_{f,p}$ and $w_{f,a}$ measure the importance of the feature $f$ on the ad and the page side correspondingly. The features used in our implementation include $tf \times idf$ weighting for unigrams, compounds based on a dictionary (e.g., "New York Times") and class labels corresponding to ad and page classification in a commercial taxonomy of about 6000 nodes, as described in [4].

## 3.2 Two-Level Process

Many different approaches have been proposed in the literature to speed up the search process [3, 8, 5], and we adopt a two-level procedure similar to [3] here.

At the first level of the search process, we check if the ads could be shown on the given page by comparing an upper bound of the ad score with $minScore_p$:

$$\text{Bid}_a \times \sum_{f \in a \cap p} w_{f,a} \times \text{maxWeight}_f > \text{minScore}_p. \quad (3)$$

In this formula, $\text{maxWeight}_f$ represents the maximum weight of the feature $f$ in any page. The value of $\text{maxWeight}_f$ is pre-calculated during index building for each feature $f$ and stored in the meta-data section of the posting list.

Using the maximal weights, the sum expression in Formula 3 is an upper bound of $Sim_{a,p}$. Thus, following from the score definition in Formula 1, the left-hand side of Formula 3 is an upper bound of $Score_{a,p}$. Using a single value for the page-side weights allows us to avoid accessing the posting lists to obtain the actual weight, as described in the following sections. The tightness of the bound depends on the variance of the weights and impacts the filtering rate at the first level of the search process. The first phase filtering does not alter the final results—we would obtain exactly the same results even if this phase is not applied.

The pages that satisfy the first level filtering are further evaluated at the second level. Here a full evaluation is performed where instead of using $\text{maxWeight}_f$ for the page feature weights, the actual page weights are retrieved from the index: we check if $\text{Score}_{a,p} > \text{minScore}_p$, and we increase the number of forecasted impressions for ad $a$ if the inequality holds.

## 3.3 WAND Operator

In this work we implement the two-level search process using a variation of the WAND operator [3]. **WAND**, standing for Weak **AND**, or Weighted **AND**, has been proposed to efficiently implement the two-level search process over large document collections. The algorithm takes as arguments a list of Boolean indicator variables $X_1, X_2, \cdots, X_k$, a list of associated positive *weights*, $w_1, w_2, \cdots, w_k$, and a threshold $h$. By definition, $\mathbf{WAND}(X_1, w_1, \cdots, X_k, w_k, h)$ is true iff

$$\sum_{1 \le i \le k} X_i w_i \ge h.$$

Given this setup, our preliminary evaluation in Formula (3) consists of evaluating for each web page $p$

$$\mathbf{WAND}(X_{f_1}, w_{f_1,a} \times \text{maxWeight}_{f_1}, X_{f_2}, w_{f_2,a} \times \text{maxWeight}_{f_2},$$
$$\cdots, X_{f_{|a|}}, w_{f_{|a|},a} \times \text{maxWeight}_{f_{|a|}}, \frac{\text{minScore}_p}{\text{Bid}_a}),$$

where $X_{f_i}$ is an indicator variable for the presence of query feature $f_i$ in web page $p$ and $|a|$ is the number of features for ad $a$. If WAND evaluates to be true, then the web page $p$ undergoes a full evaluation.

## 3.4 Forecasting Ad Impressions

The original WAND iterator [3] is used to obtain a ranked list of documents relevant to a query. To do ordinary web search using WAND, the threshold is set dynamically to the minimum score of the top results found so far as the cursors of the posting lists advance. In addition, the threshold is fixed for pages between two consecutive full evaluations. However, in ad impression forecasting, the threshold $h$ is page dependent since the lowest ranking score of shown ads, $\text{minScore}_p$, is different for each page and, for a given page, $h$ does not change at all no matter how the cursors move. A new WAND iterator needs to be devised to accommodate this particularity of our application.

To take advantage of the difference among lowest ranking scores of different pages, we index the pages in increasing order of $\text{minScore}_p$ (i.e., pages with smaller PIDs have lower $\text{minScore}_p$) so that it becomes more difficult to have the test ad shown when the posting list cursors move to larger PIDs, i.e., more pages that do not qualify Formula (3) are skipped, as demonstrated in our experimental results (Table 4).

Figure 2 describes our implementation of the new WAND iterator, showing the basic procedure to calculate the expected number of impressions for a test ad. The two methods in Figure 2, initializeCursors() and nextCandidate(), are described in detail in Figure 3 and Figure 4, respectively.

```
1. Function forecastImpressions(a)
2.    nImpressions ← 0
3.    initializeCursors() (see Figure 3)
4.    while (p ← nextCandidate() < LastID) (see Figure 4)
5.       Calculate Sim_{a,p} by Formula (2)
6.       if (Score_{a,p} > minScore_p)
7.          nImpressions ← nImpressions + Imp_p
8.    end while
9.    return nImpressions
```

**Figure 2: Main function that forecasts the number of impressions for test ad $a$. The ad features are used as a query to the inverted index of pages.**

The new WAND iterator is initialized by calling the method initializeCursors() depicted in the pseudo-code shown in Figure 3. The method receives as input the array of features extracted for the query ad. It sets the current web page to be considered (currentPage) to zero and for each query feature, $f$, it initializes its current posting cursor to be the first posting element in its posting list.

```
1. Function initializeCursors(features)
2.    currentPage ← 0
3.    for each f ∈ features
4.       C_f.beyond(0)
```

**Figure 3: The initializeCursors() method of the WAND iterator**

Similar to the original WAND iterator, our implementation also maintains two invariants during its execution:

1. All web pages with PID $\le$ currentPage have already been considered as candidates.

2. For any feature $f$, any web page containing $f$, with PID $< C_f$.PID, has already been considered as a candidate.

Note that initializeCursors() establishes these invariants.

After calling initializeCursors(), the algorithm repeatedly calls nextCandidate() to get the next candidate for full evaluation. The nextCandidate() method returns the next web page whose approximate score satisfies Formula (3). Web pages whose approximate scores do not satisfy Formula (3) are skipped. Figure 4 contains the pseudo-code for nextCandidate(), which repeatedly advances the individual feature cursors until it finds a candidate web page to return.

```
1. Function nextCandidate()
2.    repeat
3.       sort(features)
4.       pivotFeature ← findPivotFeature(features)
5.       if (pivotFeature = null) return (LastID)
6.       pivotPID ← C_pivotFeature.PID
7.       if (pivotPID = LastID) return (LastID)
8.       if (pivotPID = currentPage) //pivotPID tested?
9.          f ← pickFeature(features[1, ···, pivotFeature])
10.         C_f.beyond(pivotPID + 1)
11.      else
12.         if (C_0.PID= pivotPID) //Formula (3) satisfied?
13.            currentPage ← pivotPID
14.            return (currentPage)
15.         else
16.            f ← pickFeature(features[1, ···, pivotFeature])
17.            C_f.beyond(pivotPID)
18.   end repeat
```

**Figure 4: The nextCandidate() method of the WAND iterator**

The nextCandidate() method invokes three helper methods, sort(), findPivotFeature() and pickFeature(). The first helper, sort(), sorts the features in non-decreasing order of their current PIDs. And, findPivotFeature() returns *pivotFeature*—the first feature in the sorted order for which the accumulated (weighted by $w_{f,a}$) upper bounds of all features preceding it, including it, exceed the corresponding threshold, $\text{minScore}_p/\text{Bid}_a$. The last helper, pickFeature(), receives as input a set of features and selects the feature whose cursor is to be advanced.

The nextCandidate() method first sorts the query features in non-decreasing order of the PID's of their current postings using sort(). Next, calling findPivotFeature(), it computes the pivotFeature. If there is no such feature (meaning the sum of all features' (weighted by $w_{f,a}$) upper bounds is less than the threshold), the iterator stops and returns the constant LastID.

Then the *pivotPID* variable is set to the PID corresponding to the current posting of pivotFeature. If pivotPID is equal to the PID of the last web page considered (currentPage), WAND picks a feature preceding the pivot term and advances its iterator past currentPage by calling pickFeature() (Lines 8-10), the reason being that all web pages preceding currentPage have already been considered (by Invariant 1) and therefore the system should next consider a web page with a larger PID. Note that this move preserves Invariant 2.

We need to check whether indeed the sum of approximated contributions to pivotPID is greater than its threshold. There are two cases: if the current posting PID of all features preceding pivotFeature is equal to pivotPID, then pivotPID contains a set of query features with an accumulated (weighted by $w_{f,a}$) upper bound larger than the threshold and hence nextCandidate() sets currentPage to pivotPID, and returns this web page as a candidate for full evaluation (Lines 12-14). Otherwise, pivotPID might or might not contain all the preceding features, that is, it might or might not have enough contributions, hence WAND picks one of these features and advances its cursor to a location $\geq$ pivotPID (Lines 15-17).

The nextCandidate() method also maintains Invariant 1. Because we index web pages in increasing order of $\text{minScore}_p$, it is not possible for another web page whose PID is smaller than pivotPID to be a valid candidate since pivotFeature by definition is the first feature in the PID order for which the accumulated (weighted by $w_{f,a}$) upper bound exceeds the corresponding threshold. Hence, all web pages with a smaller PID than pivotPID can only contain features which precede pivotFeature, and the upper bound (weighted by $w_{f,a}$) on their scores are strictly less than the corresponding threshold (otherwise we would have a different pivotFeature). It follows that nextCandidate() maintains the invariant since currentPage is only advanced to pivotPID in the cases of success, i.e., finding a new valid candidate who is the first in the order.

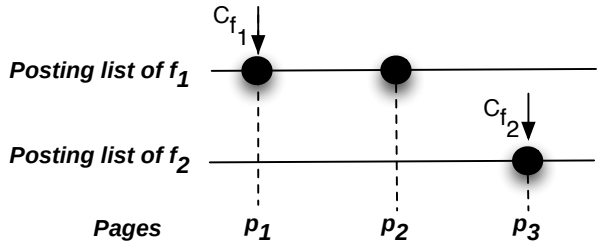


**Figure 5: A (counter) example of randomly indexed web pages ($\frac{\text{minScore}_{p_1}}{\text{Bid}_a} > w_{f_1,a} \times \text{maxWeight}_{f_1} > \frac{\text{minScore}_{p_2}}{\text{Bid}_a}$). The nextCandidate() method in Figure 4 would miss $p_2$ as a valid candidate web page.**

Note that if we indexed the web pages not in increasing order of $\text{minScore}_p$, the threshold for a page with a larger PID could be actually smaller than the one for a page with a smaller PID and Invariant 1 would not be held any more. An example of randomly indexed web pages is illustrated in Figure 5, where $\frac{\text{minScore}_{p_1}}{\text{Bid}_a} > w_{f_1,a} \times \text{maxWeight}_{f_1} > \frac{\text{minScore}_{p_2}}{\text{Bid}_a}$. The nextCandidate() method would discover pivotFeature as a feature other than $f_1$ since $\frac{\text{minScore}_{p_1}}{\text{Bid}_a} > w_{f_1,a} \times \text{maxWeight}_{f_1}$, and then miss $p_2$ as a valid candidate.

# 4. EXPERIMENTAL RESULTS

In this section, we discuss the experiments we conducted to validate our approach.

## 4.1 Data Set

We collected two weeks of actual impression events from a certain random set of hosts participating in Yahoo!'s ad network, for a total of about 1.3 billion impressions. Our evaluation uses a set of 10 million textual ads, again chosen at random from Yahoo!'s database of ads. For each distinct page $p$ in the set of impressions, we counted its total number of occurrences and determined $\text{minScore}_p$ by selecting the best matching ads as discussed in Section 2. After feature extraction, all the pages were indexed in an inverted file of features—each feature being associated with a posting list of pages where it appears.

We split the collected data set into a training set of 614 million impressions involving 318,317 distinct web pages, obtained from August 12–18, 2007, and a testing set of 699 million impressions involving 375,507 distinct web pages, obtained from August 19–25, 2007. Thus, in our experiments we forecast one week worth of impressions based on the previous week. Naturally, many pages appear both in the training set and in the test set. However, due to the

**Table 2: Average absolute relative error of forecasted impressions of *a week* at different impression levels, with the corresponding 90% confidence interval.**

| #Impressions | #Ads | Error | 90% Interval |
|---|---|---|---|
| 0-2000 | 2307 | 26364.2% | [-100%,1e5%] |
| 2001-5000 | 759 | 157.7% | [-100%,498%] |
| 5001-10000 | 768 | 69.7% | [-99.7%,196%] |
| 10001-20000 | 406 | 61.0% | [-79.7%,156%] |
| 20001-50000 | 708 | 54.6% | [-60.4%,140%] |
| 50001-100000 | 287 | 48.0% | [-54.5%,112%] |
| 100001-200000 | 186 | 32.0% | [-62.6%,69.8%] |
| 200001-500000 | 185 | 32.8% | [-60.3%,75.4%] |
| 500001-1000000 | 102 | 28.6% | [-53.3%,55.4%] |
| 1000001-10000000 | 156 | 15.7% | [-31.3%,28.8%] |

**Table 3: Average absolute relative error of forecasted impressions of *two days* at different impression levels, with the corresponding 90% confidence interval.**

| #Impressions | #Ads | Error | 90% Interval |
|---|---|---|---|
| 0-2000 | 3522 | 26212.6% | [-100%,600%] |
| 2001-5000 | 664 | 117.3% | [-99.9%,196%] |
| 5001-10000 | 417 | 108.1% | [-75.8%,121%] |
| 10001-20000 | 501 | 58.3% | [-71.1%,228%] |
| 20001-50000 | 288 | 68.3% | [-87.2%,174%] |
| 50001-100000 | 137 | 81.7% | [-65.5%,126%] |
| 100001-200000 | 133 | 60.3% | [-92.4%,91.7%] |
| 200001-500000 | 102 | 50.1% | [-64.5%,117%] |
| 500001-1000000 | 54 | 27.1% | [-46.5%,41.8%] |
| 1000001-10000000 | 46 | 23.2% | [-20.3%,41.0%] |

dynamic nature of the Web and to traffic variations, the sets are not identical and they differ both with respect to composition and to impression counts.

We evaluated our system by randomly picking approximately 6000 sample test ads. To evaluate the system on both new ads and existing ads we picked half of the sample from the 10 million set used in the training stage and the other half from another, disjoint set of ads. Note that estimating volumes for ads not previously seen is not more difficult than estimating existing ads. Preliminary results on new and existing ads separately do not show any significant difference, and we report the results on the mixed ads.

## 4.2 Effectiveness

To demonstrate the effectiveness of our approach, we compare the forecasted impressions based on the training set with the actual impressions from the test set. We split the test ads according to the forecasted impression level and report the number of sample ads at that level, the average absolute relative error and the corresponding 90% confidence interval in Table 2. The relative error is defined as

$$\text{Relative Error} = \frac{\#\text{Actual} - \#\text{Forecasted}}{\#\text{Forecasted}},$$

and the 90% interval is a range such that 90% of the test ads have a relative error in that range. (We remark in passing, that, not unexpectedly, the distribution of impressions at various levels seems to follow roughly a power law.) The average absolute relative error becomes smaller when the number of impressions becomes larger, and the confidence interval becomes tighter.

At the first glance, the errors in Table 2 seem big. However, first of all, when the traffic in the training set is identical to the test set, our method gives correct forecasting without any error. The errors come from traffic fluctuation. We conjecture that with larger periods, better forecasting could be obtained, due to less traffic fluctuation. Second, producing this forecast, even within large margins of error (which is acceptable in campaign budgeting and advertising strategy), is extremely challenging, and there is no previously published work reporting results close to ours, to the best of our knowledge. Third, one might argue that our method is only accurate for a small fraction of ads, the ones with highest volume which could be accurately estimated off-line and cached. But this is only true when the bids of those ads are fixed. When the bids change significantly, their volume would change dramatically as well. It is very common and vitally important for advertisers to optimize their investment by exploring the most cost-effective bids via methods like ours.

There is a general belief that it is easier to forecast long-term traffic rather than short-term traffic. For comparison, we show the similar results for a two day forecast in Table 3. The error indeed seem somewhat larger for short-term forecasting.

As we discussed earlier, our approach is also able to present advertisers with an impression volume vs. bid curve to help them make the most cost-effective decision on bids. To this end, note that we do not need to evaluate the same test ad repeatedly with different bid levels — instead we can evaluate the test ad once as follows: for each page, we get the minimum bid that would ensure that the test ad will be shown. We then compare these minimum bids against different bid levels to collect the forecasted impressions in a more efficient way. Several randomly selected examples of such curves (with comparison to the actual impression curves) are shown in Figure 6. Note that, as expected, the number of impressions increases monotonically as advertisers bid higher.

In Figure 6, the top row curves demonstrate that our approach can forecast impressions accurately for different shapes of curves. The curves at the bottom do not match the actual impression curves very well due to page traffic fluctuation, but share roughly the same trend or shape with the actual curves. Therefore, these curves are still very valuable in assisting advertisers to make cost-effective decisions, e.g., one advertiser might prefer to invest in an ad with a smooth bid curve, while another advertiser might prefer to bid above a "cliff" that would keep the competition away. When all the advertisers make reasonable cost-effective decisions, the traffic in ad servers could be better utilized and lead to more revenue at the end.

## 4.3 Efficiency

In order to fine-tune their bids and check the impact of wording changes, advertisers need real-time feedback. In our experiments, our approach is demonstrated to respond in sub-seconds. On an Intel Xeon 3GHz PC with 2GB RAM, with the inverted page index and the page statistics file kept in memory, the average evaluation time for a (test ad, bid) pair is 10.1 milliseconds and we require 377.1 milliseconds for curve plotting with 100 different bid levels.

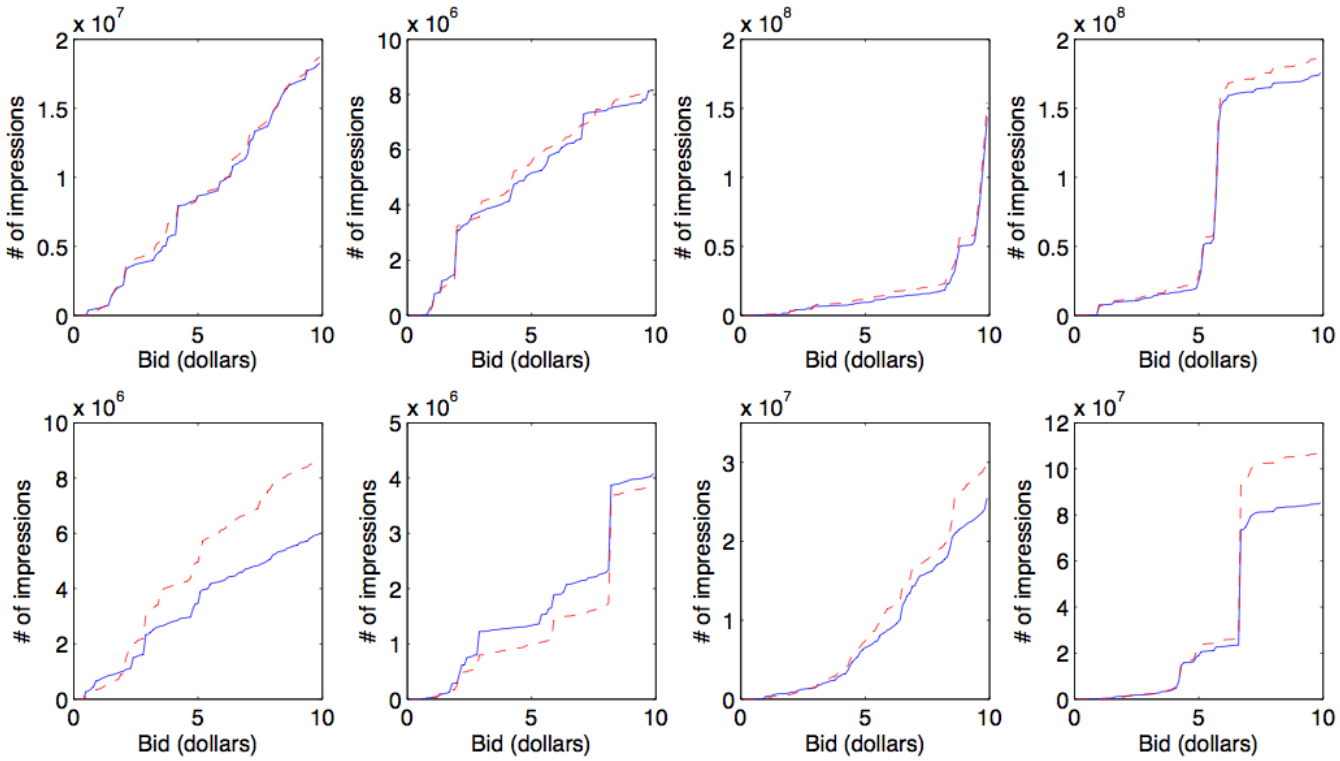


**Figure 6: The number of impression vs. bid curves (solid) of several test ads, with comparison to the actual impression curves (dashed). These curves are randomly selected from our test ads and arranged into two rows. The top row curves demonstrate that our approach can forecast impressions accurately. The curves at the bottom do not match the actual impressions very well due to page traffic fluctuation, but share roughly the same trend or shape with the actual impression curves. Thus, these curves are still very valuable to assist advertisers to make cost-effective decisions on bids.**

In Section 3, we claimed that our search skips more and more web pages as the posting list cursors advance, due to our storing of pages in increasing order of $\text{minScore}_p$. We numerically evaluate the skipping behavior here. The skipping distance is defined as the difference of PIDs of two web pages that are consecutively fully evaluated, or the difference between LastID and the PID of the last fully evaluated web page. For each test ad, we calculate average relative skipping distance with respect to the average skipping distance in the first decile of the web page index we have built. We report the mean of these statistics over all test ads for different deciles in Table 4, in which the skips are significantly larger as the cursors move towards larger PIDs. In other words, we process the last 10% of the index at rate 3000 faster than the first 10%, which means that, as long as the page index fits in memory, the forecasting time increases very slowly—our index is less than 10MB and thus we estimate that a single 8GB box can handle 25 million pages in 15 milliseconds.

**Table 4: Mean average normalized skipping distance with respect to the average skipping distance in the first decile of the web page inverted index.**

| Decile | Skips |
|---|---|
| 0%-10% | 1.00 |
| 10%-20% | 15.93 |
| 20%-30% | 32.54 |
| 30%-40% | 41.48 |
| 40%-50% | 60.35 |
| 50%-60% | 110.23 |
| 60%-70% | 252.97 |
| 70%-80% | 297.68 |
| 80%-90% | 616.31 |
| 90%-100% | 3084.45 |

## 4.4 Effectiveness vs. Efficiency Tradeoff

The usage of the upper bounds of page-to-ad scores guarantees that the proposed algorithm will find the precise count of pages $p$ such that $\text{minScore}_p < \text{Score}_{a,p}$. In our applications, due to traffic variations, precise counts are not practically usable. In this section we explore if we can trade count precision for better evaluation performance.

The performance of the two-level process depends on the tightness of the upper-bounds in the page-to-ad score estimation. This in turn depends on how well $\text{maxWeight}_f$ bounds the weight of an index posting during the scoring. Since $\text{maxWeight}_f$ is the maximum weight in the posting list, in a case of a skewed weight distribution, few postings with relatively higher weights can result in loose bounds and many unnecessary candidates passing the filter of the first phase.

To shrink the subset of web pages that are passed to full evaluation, we can enforce a stricter first level filtering at the price of allowing a few true candidates to be filtered out. This would be acceptable in our application as long as

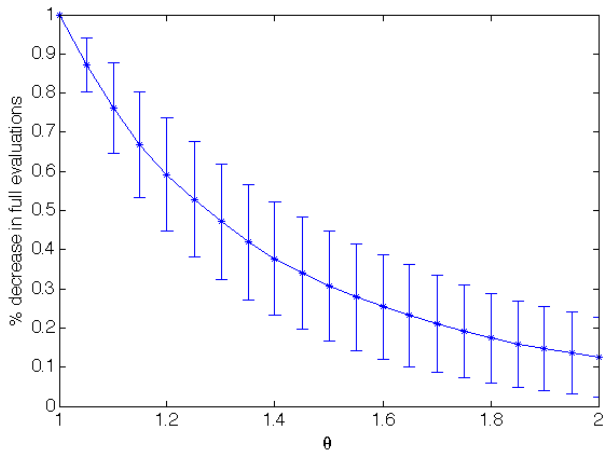


**Figure 7: (Efficiency) The average percentage decrease (with respect to $\theta = 1$) in number of full evaluations vs. $\theta$, with unit standard deviation bars. The number of full evaluations decreases quickly as the value of $\theta$ increases.**

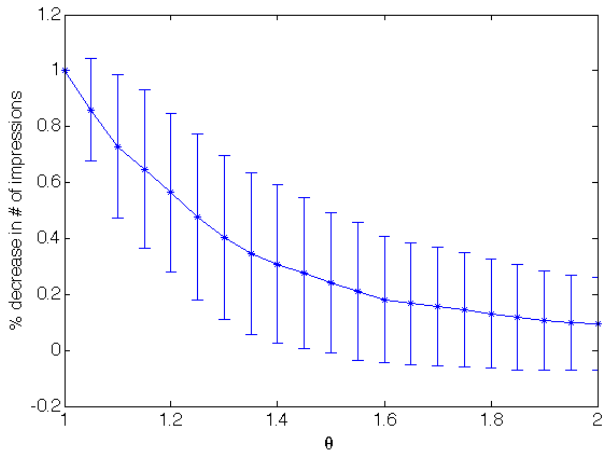


**Figure 8: (Accuracy) The average percentage decrease (with respect to $\theta = 1$) in number of forecasted impressions vs. $\theta$, with unit standard deviation bars. The forecasted impressions decrease quickly as the value of $\theta$ increases.**

the overall forecasting quality does not decrease significantly. One way to achieve this effect is to introduce an adjusting factor $\theta \geq 1$ to offset the overestimation of $\text{maxWeight}_f$, and update Formula (3) as

$$\text{Bid}_a \times \sum_{f \in a \cap p} w_{f,a} \times \text{maxWeight}_f > \theta \times \text{minScore}_p, \quad (4)$$

The term $\theta$ is related to how much increase in the forecasting error we can tolerate. When $\theta = 1$, Formula (4) becomes Formula (3). If we increase the value of $\theta$, we get a smaller candidate pool in the second phase. Simultaneously we may miss more web pages on which the test ad could actually have been shown. In other words, stricter filtering (larger $\theta$) leads to less accurate estimation but faster response. Therefore, the selection of the value of $\theta$ is indeed a tradeoff between accuracy and efficiency, as demonstrated in Figure 7 and Figure 8. As $\theta$ increases, the number of full evaluations decreases, so does the forecasted impressions. A relatively smooth curve could be fit for both Figure 7 and Figure 8. Note that even with this lossy estimation, our system is still capable to suggest bids since it captures the general trends over bid value reasonably well.

## 4.5 Evaluating Ads

To further demonstrate how our system can be used as a tool for advertisers to design and manage new ads, we manually created several ads with subtle differences[3], shown in Table 5 with their corresponding forecasted impressions. The other attributes of all the ads in Table 5 are identical, for example, they use the same description and have the same landing page (same URLs) and same bid amount.

As we can see, although most of the attributes are identical, the subtle differences make a big impact in number of impressions. It is interesting to see that the usage of terms like "cheap" and "free" *reduces* the number of forecasted impressions: the underlying system correctly predicts that such ads have lower CTRs and hence their score is adjusted downward, that is $\text{Sim}_{a,p}$ becomes lower for all pages.

[3]We could have used real ads for this experiment, but for privacy and copyright issues, we would not have been able to show the ad contents and their differences.

# 5. RELATED WORK

Current ad networks do have certain forecasting functionalities, however, unfortunately, they are not publicly accessible. One simple approach to estimating ad impressions is to run real systems with the test ads for a while to get the estimate of their performance. However, the running cost in ad servers is prohibitive, and running free test ads for a long time to alleviate sparseness is financially infeasible. Furthermore, in this naive way, advertisers have to wait a long while to get performance report of their test ads.

Another alternative to the replaying approach presented in the paper is to estimate the impressions of a new ad based on the impressions of the ads currently in the system. A machine learning framework can be built that extracts features from the ads and predicts the impressions based on ads that are similar to the test ad. While we have found no prior work on impression forecasting using this method, we believe the search based method presented in this paper has two main advantages. First, in the method proposed in this paper we use an ad-to-page matching that is the same as with the ad selection. The complexity of the ad selection mechanism can be difficult to capture by a machine learning model over the ad features. Second, an ad-to-ad comparison introduces an indirection in the data use, as the ad selection is performed between ads and pages. This additional step might introduce noise in the prediction, e.g., considering the extreme case: when the traffic in the future is identical to the past, search-based methods will give exact number of impressions without any error, and learning-based methods will highly possibly give noisy predictions.

# 6. CONCLUSIONS AND DISCUSSIONS

We presented a search-based method for ad impression forecasting of new ads (or old ads with new bids) in contextual advertising. The salient feature of our approach is that it replays the ad selection process over a log of past page

**Table 5: Several manually created DVD ads have very subtle difference in content (the other ad attributes not shown below are the same for all four ads), but the impressions of the ads differ a lot.**

| Ads | Titles | Bid Phrases | # of Impressions |
|---|---|---|---|
| 1 | Buy cheap DVDs | cheap DVDs | 1,905,455 |
| 2 | Buy great DVDs at low price | DVD | 2,398,928 |
| 3 | Free DVDs | free DVDs | 1,638,735 |
| 4 | DVDs for less | DVD | 2,423,988 |

impressions and counts how many times the new ad would have been shown if it were present in the system. In this way, during forecasting, we can employ exactly the same page-to-ad scoring as during ad serving. Ad selection is usually done based on complicated, highly tuned formulae. The proposed approach enables us to factor the scoring into the forecasting process as a black-box.

To make the search feasible, we first reduce the search over the past page impressions into a search over the unique web pages in those impressions. Then we search the space of web pages using a two-level search process: at the first level an inexpensive approximate evaluation is performed to identify candidate web pages on which a test ad could possibly have been shown; at the second level the candidates are fully evaluated and their contributions are counted.

Experimentally, we demonstrate that our approach can accurately forecast impressions of ads in the higher range of views per day. As such, it can be used by mid-size and large advertisers that run campaigns with millions of ad views per day. Besides forecasting impression counts, the approach proposed in this paper can also be used for bid suggestion, and ad evaluation. In our current work, we are exploring how to improve our estimates by combining machine learning-based methods.

## 7. REFERENCES


[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[3] A. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, pages 426–434, 2003.

[4] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 559–566, 2007.

[5] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. Maarek, and A. Soffer. Static index pruning for information retrieval systems, 2001.

[6] P. Chatterjee, D. L. Hoffman, and T. P. Novak. Modeling the clickstream: Implications for web-based advertising efforts. *Marketing Science*, 22(4):520–541, 2003.

[7] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[8] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, 2003.

[9] M. Fontoura, E. J. Shekita, J. Y. Zien, S. Rajagopalan, and A. Neumann. High performance index build algorithms for intranet search engines. In *Proceedings of the 30th Very Large Data Bases Conference*, pages 1158–1169, 2004.

[10] H. Garcia-Molina, J. Ullman, and J. Widom. *Database System Implementation*. Prentice Hall, 2000.

[11] S. Heinz and J. Zobel. Efficient single-pass index construction for text databases. *Journal of the American Society for Information Science and Technology*, 54(8):713–729, 2003.

[12] Interactive Advertising Bureau. Internet advertising revenue top $21 billion in '07, reaching record high. *http://www.iab.net/insights_research/iab_news_article/299656?o12499=*, May 2008.

[13] A. Lacerda, M. Cristo, M. A. G., W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–556, 2006.

[14] X. Long and T. Suel. Optimized query execution in large search engines with global page ordering. In *Proceedings of the 29th Very Large Data Bases Conference*, pages 129–140, 2003.

[15] S. Melnik, S. Raghavan, B. Yang, and H. Garcia-Molina. Building a distributed full-text index for the web. *ACM Transactions on Information Systems*, 19(3):217–241, 2001.

[16] B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. S. de Moura. Impedance coupling in content-targeted advertising. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 496–503, 2005.

[17] H. Turtle and J. Flood. Query evaluation: Strategies and optimizations. *Information Processing and Management*, 31(6):831–850, 1995.

[18] C. Wang, P. Zhang, R. Choi, and M. D. Eredita. Understanding consumers attitude toward advertising. In *Proceedings of the 8th Americas Conference on Information System*, pages 1143–1148, 2002.

[19] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes*. Morgan Kaufmann, 1999.

[20] W. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *Proceedings of the 15th International World Wide Web Conference*, pages 213–222, 2006.