Predicting the Click-Through Rate for Rare/New Ads

by

Kushal Dave, Vasudeva Varma

Report No: IIIT/TR/2010/15

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA

April 2010

Predicting the Click-Through Rate for Rare/New Ads

Kushal Dave
Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
kushal.dave@research.iiit.ac.in

Vasudeva Varma
Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
vv@iiit.ac.in

April 23, 2010

Abstract

Sponsored search has quickly became the largest source of revenue for Web search engines. Search engines generate revenue from click/impression events on ads. Click on an ad depend heavily on the rank at which the ad is displayed on the search page. The ordering of ads on the search page is done based on the historical click information. Hence accurately predicting the click-through rate (CTR) of an ad is of paramount importance for maximizing the revenue. We first consider the problem of removing the inherent presentation and position bias from the click-through logs for the already established ads. For newly created ads or rare ads we do not have sufficient historical information to calculate their CTR values. We present a model that inherits the click information of rare/new ads from other frequent ads which are semantically related. The semantic features are derived from the query ad click-through graphs and advertisers account information. We use gradient boosted decision trees (GBDT) as a regression model. Experiments show that the model learned using these features gives a very good prediction for the CTR values of the ads. Improvements obtained in the paper are found significant at 99% significance level.

1 Introduction

The traffic to web search engine has increased manyfolds in the recent years with most of the users starting their internet browsing from a search engine. Sponsored search is the prime source of revenue for search engines. Even with the economic downturn, sponsored search revenues summed up to 2.7$ billion in the second quarter up by 3% from the same quarter in 2008 [6]. It accounted for 44% of total revenue generated in the quarter and is expected to grow bigger in coming years.

Sponsored search can be seen as an interaction between three parties - Search engine, User and the Advertiser. The user issues a query to a search engine related to the topic on which he/she seeks information. Advertisers and search engines try to exploit the immediate interest of user in the topic by displaying ads relevant to the query topic. In a typically setting, advertisers bid on certain keywords known as bid terms and their ads may get displayed based on the match between bid term and the user query. Search engine try to rank the ads in a way that maximizes its revenue.

Search engines usually adopt one of the following pricing schemes: Pay-per-Click (PPC), Pay-per-impression (PPI), and Pay-per-Transaction (PPT). In PPC model, advertiser pays some amount each time a user clicks their ad. In PPI model, the advertiser pays every time its ad is displayed on the search page. While in PPT model the advertiser has to pay only when a user does a transaction after clicking on the ad. Among all models the PPC is predominantly used by all the modern search engines.

Earlier, Search engines used to rank ads solely based on the amount bid by the advertiser. This, intuitively, was the most obvious way of maximizing revenue. But, search engines soon realized that not all the top bided ads are relevant in the context of search query. Irrelevant ads can result in user dissatisfaction. Hence, search engines started ranking ads as a function of both relevance and expected revenue. Ads displayed in such a way caters to the interest of the user, advertiser and the search engine. Displaying ads on the search page is typically done through a two-step process. First, the top k ads are fetched from the ad database based on the extent to which they match the query context. Fetching the initial top k ads based on the content match ensures that the ads to be displayed are relevant to the query context. Once these top ads are retrieved they are ranked so as to maximize the overall expected revenue. Ranking in such a two-step fashion cater to the need of all the three parties User, Advertiser and Search Engine.

In order to maximize the expected revenue, the search engine must predict the probability of a click of an ad, more commonly known as click-through rate (CTR) of an ad. Historical click-through log is the most obvious proxy for estimating the CTR of the ads. However for new ads entering into the system and infrequent/rare ads it is very difficult to estimate the CTR as there is very little or no information available through the click-through logs. We propose a method to predict the CTR for the infrequent or new ads. We show that the CTR of the ad can be accurately predicted by inheriting it from similar ads.

The similarity features are derived from information sources such as Query-Ad click graphs, advertisers past history. We also experiment with the lexical match features between the query and ad and show that they too contribute to model.

2 MOTIVATION & SCOPE OF WORK

In a bid to maximize the revenue and to serve relevant ads to the user the search engine ranks the ads based on the expected revenue generated from an ad.

$$\epsilon_{ad}(Revenue) = Bid * Relevance_{ad}$$

One simple way to estimate the relevance of an ad is to use the number of clicks as a measure. Intuitively more the number of clicks for a query-ad pair more relevant is the ad for the query. The problem with this approach is that not all ads have the same number of impressions. Hence in order to have a common measure of comparison across all ads, the clicks are normalized by the number of impressions of the ad for a particular query i.e. the CTR of the ad. The raw CTR values calculated this way are contaminated by the position bias. Joachims et al. [11] show that the position (rank and page number) at which the ad is displayed has a dramatic impact on the click through rate of an ad. In Section 4 we describe a remedy to this situation to make the CTR values position independent so that it can be compared across ads. Ads need not always be ranked by CTR, some other measure that can accurately reflect the relevance of an ad can be used. In this paper, we refer to any such relevance ordering as CTR ordering.

The submission of ads on the ad engines is a dynamic process where the advertisers register new ads daily. In addition to the ad, CTR of an ad is also a function of the query. That is, an ad can have a different CTR for different queries. Queries to a search engine follow a power-law distribution [4], which means the tail of the query distribution is very long. Hence the probability that the search engine has to display ads for new queries is fairly high. Also, there are ads for which the system does not have sufficient click information as the ad might have recently entered the system. Ranking these new/rare ads along with ads that have been in the system for a long time is a challenging task. To solve this problem the system needs to accurately predict the CTR for these new ads. An error in prediction would result in a non-relevant ad being displayed on the results page. This not only means loss in revenue but may also degrade the user experience.

Fain and Regelson [17] predict the click-through rate for rare or novel terms by using hierarchical clustering of bid terms. The CTR of a term is predicted based on the CTR's of terms in the same cluster and the parent clusters. Apart from the term clustering information there is a wealth of information that can be used while predicting the CTR values, that can be used. Richardson et al. [18] predict the CTR values based on the information derived from the ad text such as title, body, bid term, display URL etc to predict the CTR values. The CTR of an ad is a function of the ad and the query. For an ad its CTR can be different for different queries. To account for this difference it is important to also consider the query associated with the ad in predicting the CTR value. Dembczynski et al. [12] showed that decision rules can also be used as a predicting algorithm for CTR values. They could not experiment with some important features such as ad text etc due to their unavailability in the dataset. Ashkan et. al[3] use queries intent to estimate ad's click through rate. All these previous approaches only considered features related to the ad text. Other information sources such as the query, Ad-query click graphs, queries etc. that can be used for estimating/predicting the relevance.

In a nutshell, following are the contributions of this paper: (1) we make use of the ad-query click graphs and the advertisers information to find ads similar to the new ad and show that CTR for new ads can be learned with a good accuracy from similar ads fetched from these sources. (2) We also experiment with the lexical match features between a query and an ad and find that these features help in reducing the prediction error when used with the similarity features. (3) We experiment with the gradient boosting algorithm and apply it to the problem of learning the CTR values of the ad.

The rest of the paper is organized as follows: Section 3 describes the dataset used for our experiments. In Section 4, we explain how CTR for the known ad is modeled for presentation bias. Section 5 describes the regression model used in our experiments. In Section 6, we explain various rfeatures and present the results. Section 7 deals with the models parameter optimization. We analyse the results in Section 8. Finally, we present the conclusion and future work in Section 9.

3 DATASET

The dataset used in our experiments comprised 12 days search log from a popular search engine. The data was collected for the US market. In addition to query-ad click information, the search log contains a lot of redundant fields pertaining to organic search results such as list of URLs displayed on the search page, clicked URLs etc. We filtered out all the redundant fields and only keep the required fields. The format for the entries in the log is as shown in Table 1.

Most of the fields are self explanatory. Fields from row no. 2-6 (Term id to Creative id) make a unique ad in the ad database. More about the ad hierarchy is described in section 6.1.3. The term id is a unique pointer to the term bid by the advertiser for that ad. The creative id points to the ad text. An ad in our dataset consists of following fields:

- **Bid term**: The term bid by the advertiser for the ad. This is invisible to the user but can be seen as the summary of ads in few words.
- **Title**: The title of the Ad. This and the rest of the following fields can be retrieved using the creative ID.
- **Abstract**: The abstract is the text displayed below the title. It is also known as creative text.

Table 1: Format of the Click-through log used in our study.

Query	The Query text used by the user.
Term ID	Unique Id for the bid term of the ad.
Creative ID	Unique Id pointing to the Ad text.
Adgroup ID	Points to the ad-group of the ad.
Campaign ID	Points to the Campaign of the Ad
Account ID	Advertisers unique account Id.
Clicks	Number of clicks for the query-ad pair
Impressions	Number of impressions for the query-ad pair
CTR	Normalized CTR for the query-ad pair

- **Display URL**: The URL displayed in the ad. Due to presentation issues, the display URL usually is different from that of the landing page URL. We restrict our study to the text content of ad and hence we do not consider the landing page text/URL in our model.

We consider each query-ad combination as a unique ad. This is because ads show different click behavior for different queries. For example an ad for a Sydney Sheldon book will have a different click through rate for the query "fiction books" as compared to the query "Sidney Sheldon". Hence the CTR values in our dataset are calculated for each query-ad combination.

In order to draw convincing conclusions, we only keep those ad-query pairs that either have more than 40 clicks or have at least 150 impressions. The choice of this constraint was to strike a balance between having ads in the database whose CTR values and not overly biasing the dataset by only considering query-ad pairs having more clicks. We generated a list of users that consistently have more than 100 clicks per day. These users are suspected robots clicking on the results. We removed all such entries from the dataset. Also, search queries that did not have any ads shown were filtered out. We use Pig Latin [14] scripts running on Hadoop's[1] Map Reduce framework to process the click-through logs.

After this process, we got 1,447,543 unique query-ad pairs from the click through logs. It contained 1,97,080 unique queries and 9,43,431 unique ads. We randomly divide this dataset into 65-25-10 ratio for training, testing and validation respectively. The entries in the test set are treated as if they are new/rare ads and our model predicts the CTR values for these ads. We then compare the predicted values with observed values to measure the accuracy of the model.

4 MODELING PRESENTATION BIAS

It is no secret that ads shown at the top positions get more user attention compared to lower ranked ads. Joachims et al. [11] show that user tends to

[1]http://hadoop.apache.org/

look more often at the top results compared to the lower ranked results. Hence the click-information from the past logs is contaminated with the position bias. In addition to position bias, the probability of click on an ad is also affected by the alternative ads on the page (presentation bias). In order to compare the CTR values across ads we need to alleviate this bias from the CTR values.

There are various approaches suggested in the literature for normalizing the CTR value of the ads [7, 8, 5]. We adopt an approach similar to the one described in [7]. The model assumes that the user traverses the ad list sequentially from top to bottom. This model, unlike other sequential models [5], takes the relevance of other ads into account. It makes following assumptions: (1) If the user has clicked on an ad, then he examined the ad and was attracted towards it. (2) After clicking, if the user is satisfied with the ad then he stops examining the following ads. (3) If the user is not satisfied then he/she comes back to examine other ads. Following can be inferred from the model: If the user did not click any ads then he/she did not look at the ads. If the user clicked on the ith ad then he examines all the ads from position 1 to i. If the click on the ith ad was the last click then the user stopped examining further ads below the ith ad. If there are clicks on ad after position i, then user was not satisfied by ad at position i and she continued examining the ads till the last ad.

The notion of user satisfaction introduces two types of relevance - Perceived relevance and Actual relevance. Perceived relevance can be understood by a click event, that is the user examined the ad and was attracted towards it. It is the probability that the ad is clicked given it was seen by the user, i.e. $P(Click|Seen)$. Actual relevance can be understood by a satisfaction event, that is the user the probability that the user is satisfied given that he has clicked the ad, $P(Satisfied|Seen)$. This model is different from the other models in which a click is considered as a satisfaction event, which is not always true. The actual relevance tries to model the user satisfaction.

$$\begin{aligned} P(R) &= P(Satisfied|Seen) \\ &= P(Satisfied|Click)P(Click|Seen) \end{aligned} \tag{1}$$

Where the first term models the actual relevancem while second term models the perceived relevance. Chapelle et. al [7] have shown that this model performs better than the cascade model and other sequential models. P(R) provides an unbiased estimate of the relevance of an ad from the click through logs. The CTR/relevance values, in our case are nothing but this P(R) as described above.

To calculate the CTR values, we considered click through logs of 21 days (this includes the period of 12 days from which our dataset was generated). This is done so as to have a good confidence on the CTR values used in our dataset. The CTR values for the ads in our dataset are calculated from the click through log as follows: We first divide the click through log into sessions. For each session, we get the query, list of ads displayed on the page and the list of clicked ads. We discard sessions that do not have any ad clicks, as the model assumes that the user did not look at the ads if he did not click on any of the ads. We could have assigned some probability (say 0.1) to the ads that the user examined the ads but did not click them. But we simply discard such sessions. Next, for each query-ad pair in our dataset we calculate the perceived and actual relevance of the query-ad pair over all the sessions. In addition, we also apply a beta-prior on the perceived and actual relevance values with $\alpha = 0.25$ and $\beta = 1.0$ for smoothing [7]. P(R) is then calculated as shown in equation (1).

5 MODEL

In a regression model one has a set of input features $\mathbf{x} = x_1, x_2, \cdots, x_n$ and an output or response variable y. Given a set of training samples (x_i, y_i) the model finds a function F(x) which maps input feature x to response y such that the expected value of some loss function L(y, F(x)) over the entire (training and testing) values of x,y is minimized. In our case, the response is the CTR of an ad-query pair is required to be predicted given a set of features. We use Gradient boosted decision trees (GBDT) as a regression model for predicting the CTR values. A regression tree is like a decision tree where, for each input x, the feature values for x are tested for conditions at the internal nodes of the tree and based on the feature values of x, it reaches one of the terminal node that gives the response F(x) for x. Mathematically, regression trees recursively partition the input feature space R^N into disjoint cells $R_i, for\ i = 1 \cdots I$ with each cell outputing a prediction $f_i(x)$ for all the inputs x that fall in the region Ri. The regression tree can be represented as

$$H(\mathbf{x}, \mathbf{a}) = \sum_{i=1}^{I} f_i(j) I(x \in R_i)$$

Where, a is a set of parameters for the tree and I is the identity function. GBDT consists of an ensemble of such regression trees computed in a forward step-wise manner [10]. It approximates F(x) by an additive expansion of the form

$$G(x) = \sum_{m=0}^{M} \beta_m H(\mathbf{x}, a_m)$$

At each iteration the parameters a_m and β_m are fitted to the residuals of the previous step,

$$(\beta_m, a_m) = \arg\min_{\beta, a} \sum_{i=1}^{M} L(y_i, G_{m-1}(x_i) + \beta H(x_i, a))$$

Where,

$$G_m(x) = G_{m-1}(x) + H(x, a_m)$$

One important feature of GBDT is that it outputs decision trees which are human readable. Hence these decision trees can be used to understand the click-through behavior of the ads and also can be used by ad recommendation systems to improve the quality of ads. In addition, GBDT also provides with feature importance, based on the features ability to minimize the loss function along each feature split. This information can be useful in analyzing the feature contributions towards minimizing the loss and predicting the response CTR value.

The GBDT version we use minimizes the least square loss $(y - F(x))^2$. Other loss functions such as least absolute deviation ($|y - F(x)|$) and Huber loss function can also be used [10]. In our experiments we only use least square loss. GBDT has two important optimization parameters - the number of trees and best first nodes, which sets the number of nodes for best-first search in tree growing. We experiment with these parameters to find their optimal value. As explained in Section 7, we find that by keeping the number of trees to 600 and the number of nodes for best first search to 150 gave the least error for most of the features hence we present results of the experiments with these value of the parameters.

We use the Root mean square error (RMSE) as our primary measure of performance. This metric is the root of the mean squared error between the models predicted CTR value and the actual (or observed) CTR value. All the performance improvements are reported for this metric. Our baseline model is the average CTR value on the dataset used. As in [18], we also use KL-divergence as the other performance measure. All the feature categories explained in the following section always had one ‘always on’ feature set to 1. All the results presented in this paper are statistically significant at 99% significance level. We use paired t-test for testing statistical significance. Sanderson et. al [19] show that t-test have lower error rates as compared to other significance tests.

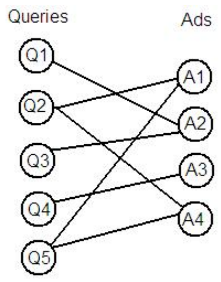
6 Features

This section describes the various similarity and syntactic features used by our GBDT model.

6.1 Similarity features

In this section, we answer the question - how much does similarity measures derived from various sources help predicting the CTR of ads. These features are based on the semantic relations of the queries and ads with other similar queries and ads. Regelson [17] have shown that similar ads (bid terms in their case) follow similar CTR distribution. The idea here is to learn the CTR values of query-ad pair from semantically similar queries and ads. We derive the semantic similarity from the query ad click-through graphs and advertiser’s information. Our click graph is built from 12 days query log (same period from which we generated our dataset). Simple normalization were performed on the query such as stop word removal, removing special characters etc. before generating the click graph.

Figure 1: Sample of Query-Ad click through Graph.



6.1.1 Similar queries (SimQ)

For rare queries we do not have sufficient information about the ads displayed and clicked for it, the CTR values for such query-ad pairs can be predicted using the CTR of other similar queries.

Figure 1 depicts a small sample of a typical query-ad click through graph. Click through graphs have been widely used for various research purposes in information retrieval such as query rewriting, query suggestion, query clustering, document clustering, collaborative filtering for ads etc. [1, 2, 9, 13].

The idea here is to represent queries as vectors and compare the query vectors to find similarity amongst the queries. A query q is represented as a vector of transition probability from q to all the ads in the graph using click frequency-inverse query frequency (CF-IQF) model similar to the one used in [9]. The transition probability from a query to an ad, $P(a_j|q_i)$ is calculated as follows: First, we calculate cf-iqf value for a query-ad pair,

$$cfiqf(q_i, a_j) = c_{ij} * iqf(a_j)$$

Next, the transition probability is calculated by normalizing the cf-iqf value for the query-ad pair by the cfiqf of the query over all the ads.

$$P(a_j|q_i) = \frac{cfiqf(q_i, a_j)}{cfiqf(q_i)}$$

Each query is represented as $\mathbf{q} = (P(a_1|q_i), P(a_2|q_i),$
$\cdots, P(a_n|q_i))$. The similarity between two queries $\mathbf{q_i}$ and $\mathbf{q_j}$ is the cosine similarity between the two query vectors.

$$Sim(\mathbf{q_i}, \mathbf{q_j}) = Cosine \frac{\vec{q_i} * \vec{q_j}}{\parallel q_i \parallel \cdot \parallel q_j \parallel}$$

This similarity score $Sim(q_i, q_j)$ can be used effectively to predict the CTR values of the query-ad pairs where we do not have much click information for the query. For each query q in our dataset, we retrieve top k queries similar q' and calculate the weighted average of the CTR values for all the ads over query q'.

$$QCTR(q,a) = \frac{\sum_k CTR(q_k) * Sim(q_i, q_k)}{\sum_k Sim(q_i, q_k)}$$

Ideally, we would have liked to use $CTR(q_j, a)$ while calculating QCTR value as is done in [1], which would have given us more focused estimate of the original CTR. But with the click-through graph we had, there were some query ad pairs (q', a) for which we did not have the CTR(q', a), hence we settled for CTR(q').

We use the $QCTR$ value as a feature in our model. We also keep the number similar queries retrieved for an ad-query pair as a feature (N_q). In addition we keep $log(QCTR+1)$ and $log(N_q+1)$ as features. We refer this set of features as SimQ. The results are as shown in Table 2. As can be seen, the SimQ features achieve a performance improvement of 18.6% over the baseline. The baseline uses only one feature which is simply the average CTR over the entire training set. These results, however, should not be directly compared to that of [18], because - we consider the CTR of an ad as a function of ad and query both, while they only consider it as function of the ad. The data filtering and the dataset generation process is also different.

Table 2: Similarity features derived from Click through logs

Feature	RMSE ($*$ e3)	KL Diver- -gence ($*$ e1)	% Improvement
Baseline	7.20	1.72	-
Sim-Q	5.86	1.42	18.61%
Sim-A	6.31	1.53	12.36%
Sim-QA	**5.68**	**1.38**	**21.11%**

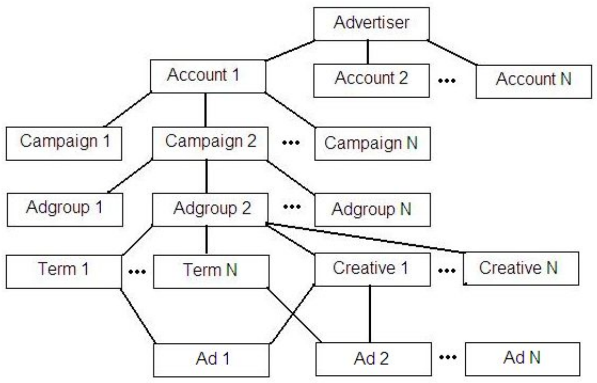
6.1.2 Similar Ads (SimA)

The query-ad click graph can be further used by inheriting CTR values of ads from other similar ads. Similarity for ads is calculated similar to the above approach. The ads are represented by vectors, where each element of vector is the transition probability from an ad to a query represented as $P(q_j|a_i)$. Formally, $\mathbf{a_i} = (P(q_1|a_i), P(q_2|a_i), \cdots, P(q_m|a_i))$, The similarity $Sim(a_i, a_j)$ is the cosine similarity between the vectors $\mathbf{a_i}$ and $\mathbf{a_j}$. The CTR of an ad is estimated as follows,

$$ACTR(a_i) = \frac{\sum_k CTR(a_k) * Sim(a_i, a_k)}{\sum_k Sim(a_i, a_k)}$$

This ACTR value is used as a feature in or model. Besides this, we also take the number of similar ads retrieved (N_a) as one of the feature. Also, as done above we take $log(ACTR+1)$ and $log(N_a+1)$ as features. In this paper, this features are referred as SimA. The results are shown in table 2. The SimA features achieve an 12.36% error reduction compared to the baseline. When we combined the SimQ and SimA features (referred as SimQA features) the accuracy of the model improves further and 21.11% error reduction is achieved.

Figure 2: A typical Ad hierarchy maintained by ad engines



6.1.3 Ad Hierarchy (AdH)

Advertisements on an ad engine are typically maintained in some kind of a hierarchy. One hierarchy of such kind is as shown in the Figure 2. There are numerous reasons for maintaining ads in a hierarchy - (1) Advertiser's business may span various business units. For example, Amazon has an online marketplace for books, CDs etc and it also deals with selling cloud computing services to the users. These two areas are entirely different business units. Ads from the same advertiser but from different business units are maintained in different accounts. (2) For each business unit, the advertisers can have ads on a range of products. Advertisements from the same account on similar products fall under the same Campaign. Ads in same campaign usually are focused towards one product or similar products. (3) Adgroups do further granular classification of ads. (4) Finally, an ad comprises a bid term and ad text (creative). Combination of these two makes an ad. The reason behind keeping the term id and creative id separate is that different ads may share the same bid term or the same creative hence it is easier to maintain it in a hierarchy. Such ad hierarchies can be exploited to get good insights on click through rate of new ads coming from already established accounts. The idea here is that ads coming from the same group usually talk about same or similar products. Hence we find other ads grouped in the same category at each level and use the CTR of these ads as features.

Table 3: Features computed from the Ad-hierarchy

Feature	RMSE (1e-3)	KL Diver- -gence ((1e-1)	% Improvement
Baseline	7.20	1.72	-
Term	6.24	1.45	13.34%
Creative	6.51	1.50	9.6%
Adgroup	5.87	1.35	18.48%
Campaign	**5.67**	**1.32**	**21.25%**
Account	5.94	1.39	17.50%
AdH	6.20	1.46	13.9%
SimQA + Camp	**5.28**	**1.24**	**26.67%**

We aggregate ads at each level viz. Term, Creative, Adgroup, Campaign and Account, compute the average within each group and use them as features in our model. For all the ads in the dataset, the average CTR was computed over the training set. These average values are smoothed using the average CTR before using them as features.

We also use the number of similar ads in the category as a feature. In addition we also compute log(f+1) of these features f and use them into our model.

There are some important advantages of using such hierarchy for CTR prediction, like - these features can be used as a good proxy for the advertiser's profile, reputation etc. As the ad hierarchy is already in place, estimating these within group CTR values does not require huge computational effort.

The results for AdH features are as shown in table 3. Of all the features, Campaign gave the least error, it achieved a handsome 21.25% error reduction. Aggregating CTR values at the creative level did not yield much improvement. One reason for it can be that not many ads have the same creative (ad text), hence the information at Creative level is very sparse. Combining all the features together (AdH), surprisingly, did not give any better results compared to most of the individual categories. After combining the SimQA and the Campaign features, the model achieved 26.67% error reduction.

One important thing to note during our experiments was that for some of the advertisers the average CTR values were very close for Adgroup, Campaign and Account categories. When we investigated the cause, the dataset had some ads coming small business enterprises. Usually, these advertisers only have a single account and only a single category of products on which they advertise. For ads coming from such advertisers, the average CTR values were similar.

6.2 Query-Ad similarity

Finally, we also wanted to compare the performance of the similarity based features with the traditional query-ad syntactic match features and to assess the

Table 4: Performance of Query-ad lexical overlap features

Feature	RMSE (1e-3)	KL Diver- -gence (1e-1)	% Improvement
Baseline	7.20	1.72	-
QADL	6.50	1.56	9.72%
SimQA+Camp +QADL	**5.14**	**1.21**	**28.61%**

performance of model using these two sets of features together. The principle behind such similarity features is: More the lexical overlap, more relevant is the ad to a query and hence higher is the CTR for such ads. Recently there has been a lot of research done in the area of query-ad text match [16, 20, 15]. Raghavan et. al [16] use various lexical overlap measures to classify ads as relevant or non-relevant. Shaparenko et. al. [20] investigate various word pair indicator features and find that these features perform better than the vector space and language modeling features. In this paper, we restrict ourselves to the score based lexical models such as cosine similarity, edit distance.

In an attempt to capture how relevant an ad is to the query, we compute the lexical overlap between the query and these ad units. We compute various text matching features such as cosine similarity, word overlap, character overlap, and string edit distance for each combination of unigrams and bi-grams. Length features like query length, bid term length, and title length are also incorporated. In addition, we also include binary features like whether the bid term or the title occurred in the query. In all, this category had 38 features. We refer to this category of features as QADL.

As shown in Table 4, the QADL features alone achieve a 7.12% improvement over the baseline. When combined with the SimQA+Campaign model the performance improves further to 28.61% , approximately a 2% improvement over the SimQA+Campaign similarity model. Overall the model *SimQA+Campaign+QADL* has 50 features.

Figure 3: Varying the Ntrees parameter in GBDT

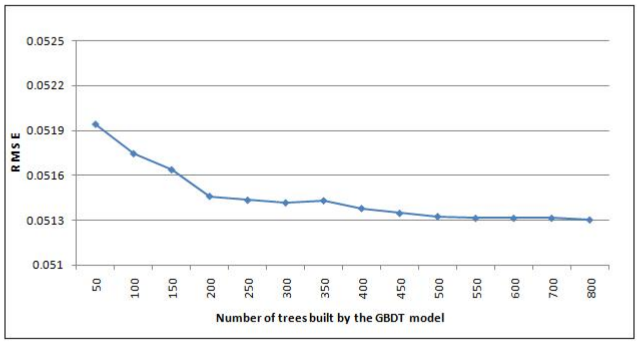


Table 5: Feature Importance: Top 10 features

Rank	Features	Category	Importance(%)
1	Campaign	AdH	100
2	ACTR	SimA	30.4657
3	No. of Similar Ads in Campaign	AdH	18.339
4	No. of Similar ads using ad similarity (N_a)	SimA	17.7344
5	Log(ACTR)	SimA	15.9063
6	Unigram word overlap Query & title of Ad	QADL	11.2283
7	Bigram word overlap between query and display url	QADL	10.432
8	QCTR	SimQ	8.10356
9	Cosine score between query and display url	QADL	6.84033
10	Log of Average CTR of ads at campaign level	AdH	6.41187

7 GBDT parameters

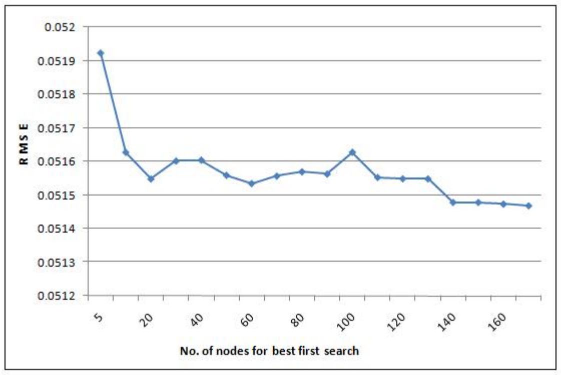
We experiment with the following GBDT parameters: The number of trees built by the model (Ntrees) and the number of nodes (Nnodes) for the best first search. These experiments have been performed on the validation set. We first varied the parameter Ntress by keeping the parameter Nnodes fixed to a default value of ten. As can be seen from Figure 3), the error reduces as we keep on increasing Ntrees till 600, after that the improvements very minor hence we stopped at 800. It is interesting to note that the model does not overfit as the number of trees increases. The average performance improvement from Ntrees=50 to Ntrees=600 was found to be 1.2%. We only report results for SimQA+ Campaign+QADL model, but all the other models showed similar trends hence we fixed the Ntree parameter to 600 for our experiments. Next, we varied Nnodes parameter by keeping the Ntrees fixed to 600. As shown in the Figure 4 the error fluctuates as we increase the value for Nnodes, before settling down to a minimum at Nnodes=150 after which there is infinitesimal improvement in the model hence we stop at 170. The average reduction in error from Nnodes=5 to Nnodes=150 was found to be 0.85%.

8 ANALYSIS & DISCUSSION

One Interesting question is - Which features contribute the most to the model *SimQA+Campaign +QADL*. The GBDT model provides feature importance [10] which is calculated by keeping track of the features ability to minimize the loss function along each feature split and then calculating the total reduction in loss for each feature. Mathematically, for each tree the relative importance is calculated as

$$\hat{P}_j^2(G) = \sum_{t=1}^{L-1} \hat{T}_j^2 I(v_t = j) \tag{2}$$

Figure 4: Varying the Nnodes parameter in GBDT



where the summation is over the internal nodes t of a L-terminal node tree G, v_t is the splitting feature associated with node t and $\hat{T}_j^2$ is the corresponding improvement in least-squared error as a result of the split. This information can be used to analyze the contribution of the features to the model. Some of the top contributing features are shown in Table 5.

The top five features come from the similarity category and the top 25 features in this list cover all the similarity features. The campaign features from AdH category contribute the most. This Adh feature can even be utilized for those advertisers who are having a fresh account, (which can be the case with small enterprises), where the Ad hierarchy does not exist or is pretty small and sparse. In such cases, the AdH features of similar ads can be used. The similarity can be computed as explained in Section 6.1.2. After campaign, the ad similarity features contribute the most. One important thing to note is that some of the features that learn from set of similar ads might have overlapping set of ads. That is, an ad whose similarity is derived from the query-ad click graph might come from the same campaign or the same ad-group. We did not take any explicit measure to ensure that these sets are disjoint, as we were also interested in assessing the individual contribution of these features. It would be interesting to see how these features perform when these similar set of ads are disjoint.

It is also interesting to note that syntactic features such as unigram/bigram word overlap between query-ad title and query-display URL also appear in the top 10 list, which tells us that the ad title and ad URL are two useful indicator for relevance of an ad. Features pertaining to match between query and the creative(short description or abstract) did not contribute much. The average contribution from such features was found to be 3.2%.

9 Conclusion & Future work

We have proposed an approach to predict the CTR for new ads based on the similarity with other ads. The similarity of ads is derived from sources like query ad click-through graph and advertisement hierarchies maintained by the ad engine. The GBDT model learns from these similarity features and gives good prediction on the CTR values for new ads. In addition, we also experimented with some of the features based on the lexical match between the query-ad units and find that combining them with the similarity features gives an additional improvement to the model. The features described in the paper are easily implementable on the web scale.

Analysis of the feature's contribution shows that the features derived from the ad hierarchy and from the click-through graphs contribute the most to the model followed by some of the word overlap features.

In our future work, we would like to explore the user behavior information for predicting CTR by considering the CTR as a function of the ad, query and the user. Yan et. al [21] show that user's past history can be used in improving the CTR by serving ads relevant to the user. These user similarity features can also be used for prediction of CTR. The notion of similar ads and queries can be extended to estimate CTR from similar users by generating query-user graph or the query-ad click graphs.

References

[1] Tasos Anastasakos, Dustin Hillard, Sanjay Kshetramade, and Hema Raghavan. A collaborative filtering approach to ad recommendation using the query-ad click graph. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1927–1930, New York, NY, USA, 2009. ACM.

[2] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. Simrank++: query rewriting through link analysis of the click graph. *Proc. VLDB Endow.*, 1(1):408–421, 2008.

[3] Azin Ashkan, Charles L. A. Clarke, Eugene Agichtein, and Qi Guo. Estimating ad clickthrough rate through query intent analysis. In *WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 222–229, Washington, DC, USA, 2009. IEEE Computer Society.

[4] Ricardo A. Baeza-Yates and Felipe Saint-Jean. A three level search engine index based in query log distribution. In *SPIRE*, pages 56–65, 2003.

[5] Hila Becker, Christopher Meek, and David Maxwell Chickering. Modeling contextual factors of click rates. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 1310–1315. AAAI Press, 2007.

[6] Internet Advertising Bureau. *Internet Advertising Revenue Report.* IAB 2009 second quarter results.

[7] Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 1–10, New York, NY, USA, 2009. ACM.

[8] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 87–94, New York, NY, USA, 2008. ACM.

[9] Hongbo Deng, Irwin King, and Michael R. Lyu. Entropy-biased models for query representation on the click graph. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346, New York, NY, USA, 2009. ACM.

[10] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, 2002.

[11] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA, 2005. ACM.

[12] W. Kotlowski K. Debmbsczynski and D. Weiss. Predicting ads clickthrough rate with decision rules. In *Workshop on Target and Ranking for Online Advertising, WWW 08*, 2008.

[13] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346, New York, NY, USA, 2008. ACM.

[14] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110, New York, NY, USA, 2008. ACM.

[15] H. Raghavan and R. Iyer. Evaluating vector-space and probabilistic models for query to ad matching. In *IA '08: in Proceedings SIGIR Workshop on Information Retrieval for Advertising, 2008.*, 2008.

[16] Hema Raghavan and Dustin Hillard. A relevance model based filter for improving ad quality. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 762–763, New York, NY, USA, 2009. ACM.

[17] Moira Regelson and Daniel C. Fain. Predicting click-through rate using keyword clusters. In *Electronic Commerce (EC)*. ACM, 2006.

[18] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 521–530, New York, NY, USA, 2007. ACM.

[19] Mark Sanderson and Justin Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169, New York, NY, USA, 2005. ACM.

[20] Benyah Shaparenko, Özgür Çetin, and Rukmini Iyer. Data-driven text features for sponsored search click prediction. In *ADKDD '09: Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising*, pages 46–54, New York, NY, USA, 2009. ACM.

[21] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 261–270, New York, NY, USA, 2009. ACM.