Learning to Advertise
How many ads are enough?

Bo Wang[‡], Zhaonan Li[‡], and Jie Tang[‡]

[‡]Department of Computer Science, Tsinghua University, Beijing, China

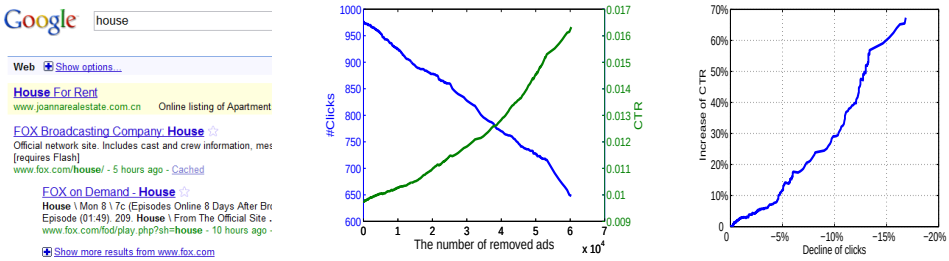
Abstract. Sponsored advertisement(ad) has already become the major source of revenue for most popular search engines. One fundamental challenge facing all search engines is how to achieve a balance between the number of displayed ads and the potential annoyance to the users. Displaying more ads would improve the chance for the user clicking an ad. However, when the ads are not really relevant to the users' interests, displaying more may annoy them and even "train" them to ignore ads. In this paper, we study an interesting problem that how many ads should be displayed for a given query. We use statistics on real ads click-through data to show the existence of the problem and the possibility to predict the ideal number. There are two main observations: 1) when the click entropy of a query exceeds a threshold, the CTR of that query will be very near zero; 2) the threshold of click entropy can be automatically determined when the number of removed ads is given. Further, we propose a learning approach to rank the ads and to predict the number of displayed ads for a given query. The experimental results on a commercial search engine dataset validate the effectiveness of the proposed approach.

1 Introduction

Sponsored search places ads on the result pages of web search engines for different queries. All major web search engines (Google, Microsoft, Yahoo!) derive significant revenue from such ads. However, the advertisement problem is often treated as the same problem as traditional web search, i.e., to find the most relevant ads for a given query. One different and also usually ignored problem is "how many ads are enough for a sponsored search". Recently, a few research works have been conducted on this problem [5, 6, 8, 17]. For example, Broder et al. study the problem of "whether to swing", that is, whether to show ads for an incoming query [3]; Zhu et al. propose a method to directly optimize the revenue in sponsored search [22]. In most existing search engines, the problem has been treated as an engineering issue. For example, some search engine always displays a fixed number of ads and some search engine uses heuristic rules to determine the number of displayed ads. However, the key question is still open, i.e., how to optimize the number of displayed ads for an incoming query?

Motivation Example. Figure 1 (a) illustrates an example of sponsored search. The query is "house" and the first one is a suggested ad with yellow background, and the search results are listed in the bottom of the page. Our goal is to predict

the number of displayed ads for a given query. The problem is not easy, as it is usually difficult to accurately define the relevance between an ad and the query. We conduct several statistical studies on the log data of a commercial search engine, the procedure is in two-stage: First, for each query, we obtain all the returned ads by the search engine; Second, we use some method to remove several unnecessarily displayed ads (detailed in Section 4). Figure 1 (b) and (c) are the statistical results on a large click-through data (DS_BroadMatch dataset in section 3). The number of "removed ads" refers to the total number of ads cut off in the second stage for all the queries. Figure 1(b) shows how #clicks and Click-Through-Rate (CTR) vary with the number of removed ads. We see that with the number of removed ads increasing, #clicks decreases, while CTR clearly increases. This matches our intuition well, displaying more ads will gain more clicks, but if many of them are irrelevant, it will hurt CTR. Figure 1(c) further shows how CTR increases as #clicks decreases. This is very interesting. It is also reported that many clicks on the first displayed ad are done before the users realize that it is not the first search result. A basic idea here is that we can remove some displayed ads to achieve a better performance on CTR.



(a) Sponsored search (b) #removed ads vs. #clicks & CTR (c) #clicks vs. CTR

Fig. 1. Motivation example.

Thus, the problem becomes how to predict the number of displayed ads for an incoming query which is non-trivial and poses two unique challenges:

- *Ad ranking.* For a given query, a list of related ads will be returned. Ads displayed at the top positions should be more relevant to the query. Thus, the first challenge is how to rank these ads.
- *Ad Number Prediction.* After we get the ranking list of ads, it is necessary to answer the question "how many ads should we show?".

Contributions. To address the above two challenges, we propose a learning-based framework. To summarize, our contributions are three-fold:

- We performed a deep analysis of the click-through data and found that when the click entropy of a query exceeds a threshold, the CTR of that query will be very near zero.
- We developed a method to determine the number of displayed ads for a given query by an automatically selected threshold of click entropy.

- We conducted experiments on a commercial search engine and experimental results validate the effectiveness of the proposed approach.

2 Problem Definition

Suppose we have the click-through data collected from a search engine, each record can be represented by a triple $\{q, ad_q(p), c_q(p)\}$, where for each query q, $ad_q(p)$ is the ad at the position p returned by the search engine and $c_q(p)$ is a binary indicator which is 1 if this ad is clicked under this query, otherwise 0. For each ad $ad_q(p)$, there is an associated feature vector $x_q(p)$ extracted from a query-ad pair $(q, ad_q(p))$ and can be utilized for ranking model learning.

Ad Ranking: Given the training data denoted by $L = \{q, AD_q, C_q\}_{q \in \mathcal{Q}}$ in which $\mathcal{Q}$ is the query collection, for each $q \in \mathcal{Q}$, $AD_q = \{ad_q(1), \cdots, ad_q(n_q)\}$ is its related ad list and $C_q = \{c_q(1), \cdots, c_q(n_q)\}$ is the click indicators where n_q is the total number of displayed ads. Similarly, the test data can be denoted by $T = \{q', AD_{q'}\}_{q' \in \mathcal{Q}'}$ where $\mathcal{Q}'$ is the query collection. In this task, we try to learn a ranking function for displaying the query-related ads by relevance. For each query $q' \in \mathcal{Q}'$, the output of this task is the ranked ad list $R_{q'} = \{ad_{q'}(i_1), \cdots, ad_{q'}(i_{n_q})\}$ where $(i_1, \cdots, i_{n_q})$ is a permutation of $(1, \cdots, n_q)$.

Ad Number Prediction: Given the ranked ad list $R_{q'}$ for query q', in this task we try to determine the number of displayed ads k and then display the top-k ads. The output of this task can be denoted by a tuple $O = \{q', R^k_{q'}\}_{q' \in \mathcal{Q}'}$ where $R^k_{q'}$ are the top-k ads from $R_{q'}$.

Our problem is quite different from existing works on advertisement recommendation. Zhu et al. propose a method to directly optimize the revenue in sponsored search [22]. However, they only consider how to maximize the revenue, but ignore the experience of users. Actually, when no ads are relevant to the users' interests, displaying irrelevant ads may lead to much complains from the users and even train the user to ignore ads. Broder et al. study the problem of "whether to swing", that is, whether to show ads for an incoming query [3]. However, they simplify the problem as a binary classification problem, while in most real cases, the problem is more complex and often requires a dynamic number for the displayed ads. Few works have been done about dynamically predicting the number of displayed ads for a given query.

3 Data Insight Analysis

3.1 Data Set

In this paper, we use one month click-through data collected from the log of a famous Chinese search engine Sogou[1], the search department of Sohu company which is a premier online brand in China and indispensable to the daily life of

[1] `http://www.sogou.com`

millions of Chinese. In the data set, each record consists of user's query, ad's keyword, ad's title, ad's description, displayed position of ad, and ad's bidding price . For the training dataset DS_BroadMatch, the total size is around 3.5GB which contains about 4 million queries, 60k keywords and 80k ads with 25 million records and 400k clicks. In this dataset, the ad is triggered when there are common words between keyword and user's search query. We also have another little training dataset DS_ExactMatch which a subset of DS_BroadMatch and contains about 28k queries, 29k keywords and 53k ads with 4.4 million records and 150k clicks. In DS_ExactMatch, the ad is triggered only when there are exactly matched words between keywords and user's search query. For the test set, the total size is about 90MB with 430k records and 1k clicks.

3.2 Position vs. Click-Through Rate (CTR)

Figure 2 illustrates how CTR varies according to the positions on the dataset DS_ExactMatch. We can see that the actions of clicks mainly fall into the top three positions of ad list for a query, so the clicks are position-dependent.

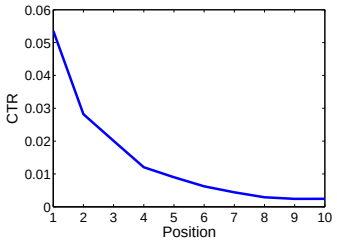


Fig. 2. How CTR varies with the positions.

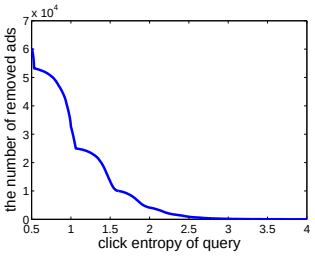


Fig. 3. How the number of removed ads varies with the click entropy of a query.

3.3 Click Entropy

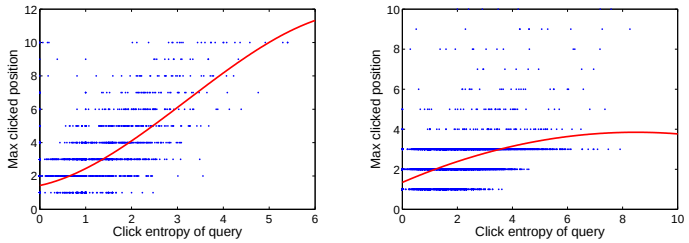
In this section we conduct several data analyses based on the measure called click entropy. For a given query q, the click entropy is defined as follows [11]:

$$ClickEntropy(q) = \sum_{ad \in \mathcal{P}(q)} -P(ad|q) log_2 P(ad|q) \quad (1)$$

where $\mathcal{P}(q)$ is the collection of ads clicked on query q and $P(ad|q) = \frac{|Clicks(q,ad)|}{|Clicks(q)|}$ is the ratio of the number of clicks on ad to the number of clicks on query q.

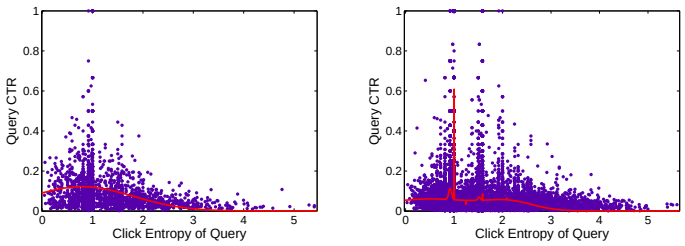
A smaller click entropy means that the majorities of users agree with each other on a small number of ads while a larger click entropy indicates a bigger query diversity, that is, many different ads are clicked for the same query.

Click Entropy vs. #Removed ads. Figure 3 shows how the number of removed ads varies with the click entropy of a query on the dataset DS_BroadMatch. By this distribution, for a query, if we want to remove a given number of ads, we



(a)DS_ExactMatch (b)DS_BroadMatch

Fig. 4. How Max-Clicked-Position varies with the click entropy on the two datasets.



(a)DS_ExactMatch (b)DS_BroadMatch

Fig. 5. How QueryCTR varies with the click entropy on the two datasets.

can automatically obtain the threshold of the click entropy which can be utilized for helping determine the number of displayed ads.

Click Entropy vs. Max-Clicked-Position. For a query, Max-Clicked-Position is the last position of clicked ad. Figure 4 shows how the Max-Clicked-Position varies with the click entropy on the two datasets. The observations are as follows:

- As the click entropy increases, the Max-Clicked-Position will be larger.
- The values of click entropy on the dataset DS_ExactMatch are in a smaller range than DS_BroadMatch which implies that when the query and keywords are exactly matched, the click actions of users are more likely consistent.
- On the dataset DS_ExactMatch, the clicked positions vary from 1 to 10, while on the dataset DS_BroadMatch, the clicked positions only vary from 1 to 4. The intuition behind this observation is that while query and keyword are exactly matched, users will scan all the ads because of the high relevance,but while query and keyword are broadly matched, users will only scan the top four ads and ignore the others. This is very interesting which implies that for a query broadly matched with the ads' keywords, we should display fewer ads than those exactly matched with the ads' keywords.

Click Entropy vs. QueryCTR. Figure 5 shows how QueryCTR varies with the click entropy of a query. QueryCTR is the ratio of the number of clicks of a query to the number of impressions of this query. We can conclude that when the click entropy of a query is greater than 3, the QueryCTR will be very near

zero. This observation is very interesting, the QueryCTR is the summation of the ads' click entropy, so we can utilize this observation to help determine the number of displayed ads for a given query.

4 Ad Ranking and Number Prediction

4.1 Basic Idea

We propose a two-stage approach corresponding to the two challenges of our problem. First, we learn a function for predicting CTR based on the click-through data by which the ads can be ranked. Second, we propose a heuristic method to determine the number of displayed ads based on the click entropy of query. For a query, the click entropy is the summation of entropy of each clicked ads, so we consider the ads in a top-down mode, once the addition of one ad leads to the excess of a predefined threshold by the click entropy, we then cut off the rest ads. By this way, we can automatically determine the number of displayed ads.

4.2 Learning Algorithm

Ad Ranking: In this task, we aim to rank all the related ads of a given query by relevance. Specifically, given each record $\{q, ad_q(p), c_q(p)\}$ from the click-through data $L = \{q, AD_q, C_q\}_{q \in \mathcal{Q}}$, we can first extract its associated feature vector $x_q(p)$ from the query-ad pair, then obtain one training instance $\{x_q(p), c_q(p)\}$. Similarly, we can generate the whole training data $L' = \{x_q(p), c_q(p)\}_{q \in \mathcal{Q}, p=1,\cdots,n_q} \in \mathbb{R}^d \times \{0, 1\}$ from the click-through data where d is the number of features.

Let $(x, c) \in L'$ be an instance from the training data where $x \in \mathbb{R}^d$ is the feature vector and $c \in \{0, 1\}$ is the associated click indicator. In order to predict the CTR of an ad, we can learn a logistic regression model as follows whose output is the probability of that ad being clicked:

$$P(c = 1|x) = \frac{1}{1 + e^{-\sum_i w_i x_i}} \tag{2}$$

where x_i is the i-th feature of x and w_i is the weight for that feature. $P(c = 1|x)$ is the predicted CTR of that ad whose feature vector is x.

For training, we can use the maximum likelihood method for parameter learning; for test, given a query, we can use the learnt logistic regression model for predicting the CTR of one ad.

Ad Number Prediction: Given a query q, we can incrementally add one ad to the set of displayed ads in the top-down mode, and the clicked ads will contribute to the click entropy. We can repeat this process until the click entropy exceeds a predefined threshold, and then stop. By then, the size of that set is exactly the number of the displayed ads for that query.

It is also worth noting that how to automatically determine the threshold of click entropy. Figure 3 demonstrates that when the click entropy of a query exceeds 3, the QueryCTR of that query will be very near zero. According to

Algorithm 1: Learning to Advertise

Input: Training set: $L = \{q, AD_q, C_q\}_{q \in \mathcal{Q}}$
Test set: $T = \{q', AD_{q'}\}_{q' \in \mathcal{Q}'}$
Threshold of click entropy: η
Output: the number of displayed ads k and $O = \{q', R_{q'}\}_{q' \in \mathcal{Q}'}$

Ad ranking:
Function learning for predicting CTRs from L
$$P(c = 1|x) = \frac{1}{1+e^{-\sum_i w_i x_i}}$$

Ad Number Prediction:
for $q' \in \mathcal{Q}'$ **do**
  Rank $AD_{q'}$ by the predicted CTRs $P(c = 1|x)$
  Let the number $k = 0$ and click entropy $CE = 0$
  while $CE \leq \eta$ **do**
    $k = k + 1$
    if $ad_{q'}(k)$ is predicted to be clicked
      $CE = CE - P(ad_{q'}(k)|q')log_2 P(ad_{q'}(k)|q')$
    end if
  end while
  $R_{q'} = AD_{q'}(1 : k)$
  Output k and $O = \{q', R_{q'}\}_{q' \in \mathcal{Q}'}$
end for

this relationship, we can learn a fitting model(eg. regression model) from the statistics of data, then for a given number of ads to be cut down, we can use the learned model to predict the threshold of click entropy.

The method can also be applied to a new query. Based on the learned logistic model, we can first predict the CTR for each ad related to the new query [17], then predict the number of ads based on the click entropy for the new query.

4.3 Feature Definition

Table 1 lists all the 30 features extracted from query-ad title pair, query-ad pair, and query-keyword pair which can be divided into three categories: Relevance-related, CTR-related and Ads-related.

Relevance-related features. The relevance-related features consist of low-level and high-level ones. The low-level features include highlight, TF, TF*IDF and the overlap, which can be used to measure the relevance based on keyword matching. The high-level features include cosine similarity, BM25 and LMIR, which can be used to measure the relevance beyond keyword matching.

CTR-related features. AdCTR can be defined as the ratio of the number of ad clicks to the total number of ad impressions. Similarly, we can define keyCTR and titleCTR. KeyCTR corresponds to the multiple advertising for the specific keyword. And titleCTR corresponds to multiple advertising with the same ad title. We also introduce features keyTitleCTR and keyAdCTR, because usually the assignment of a keyword to an ad is determined by the sponsors and the search engine company, the quality of this assignment will affect the ad CTR.

Ads-related features. We introduce some features for ads themself, such as the length of ad title, the bidding price, the match type and the position.

Table 1. Feature definitions between query and ads.

Category	No.	Feature Name	Feature Description
Relevance	1	Highlight of title	ratio of highlight terms of query within title to the length of title
	2	Highlight of ad	ratio of highlight terms of query within ad title+description to the length of ad title+description
	3	TF of title	term frequency between query and the title of ad
	4	TF of ad	term frequency between query and the title+description of ad
	5	TF of keyword	term frequency between query and the keyword of ad
	6	TF*IDF of title	TF*IDF between query and the title of ad
	7	TF*IDF of ad	TF*IDF between query and the title+description of ad
	8	TF*IDF of keyword	TF*IDF between query and the keyword of ad
	9	Overlap of title	1 if query terms appear in the title of ad; 0 otherwise
	10	Overlap of ad	1 if query terms appear in the title+description of ad; 0 otherwise
	11	Overlap of keyword	1 if query terms appear in the keywords of ad; 0 otherwise
	12	cos_sim of title	cosine similarity between query and the title of ad
	13	cos_sim of ad	cosine similarity between query and the title+description of ad
	14	cos_sim of keyword	cosine similarity between query and the keywords of ad
	15	BM25 of title	BM25 value between query and the title of ad
	16	BM25 of ad	BM25 value between query and the title+description of ad
	17	BM25 of keyword	BM25 value between query and the keywords of ad
	18	LMIR of title	LMIR value between query and the title of ad
	19	LMIR of ad	LMIR value between query and the title+description of ad
	20	LMIR of keyword	LMIR value between query and the keywords of ad
CTR	21	keyCTR	CTR of keywords
	22	titleCTR	CTR of the title of ad
	23	adCTR	CTR of title+description of ad
	24	keyTitleCTR	CTR of keyword+title of ad
	25	keyAdCTR	CTR of keyword+title+description of ad
Ads	26	title_length	the length of title of ad
	27	ad_length	the length of title+description of ad
	28	biding_price	biding price of keyword
	29	match_type	match type between query and ad (exact match, broad match)
	30	position	position of ad in the ad list

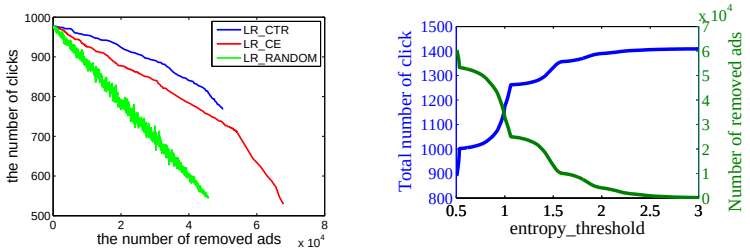
5 Experimental Results

5.1 Evaluation, Baselines and Experiment Setting

Evaluation. We qualitatively evaluate all the methods by the total number of clicks for all queries in the test dataset: $\#click(q) = \sum_{p=1}^{n_q} c_q(p)$.

For evaluation, we first remove a certain number of ads for a query in the test dataset by different ways, and then find the way which leads to the least reduction of the number of clicks.

Baselines. In order to quantitatively evaluate our approach, we compare our method with two other baselines. Assume that we want to cut down N ads in total. For the first baseline LR_CTR, for each query in the test dataset, we predict the CTRs for the query-related ads, and then pool the returned ads for all the queries and re-rank them by the predicted CTRs, finally remove the last N ads with lowest CTRs. The major problem for LR_CTR is that it cannot be updated in an online manner, that is, we need to know all the predicted CTRs for all the queries in the test dataset in advance. This is impossible for determining the removed ads for a given query. For the second baseline LR_RANDOM, we predict the CTRs of the query-related ads for each query in the test dataset, and then only remove the last ad with some probability for each query. We can tune the probability for removing a certain number of ads, the disadvantage



(a) (b)

Fig. 6. (a) How the total number of clicks varies with the number of removed ads for all three methods; (b) How the total number of clicks and the total number of removed ads vary with the threshold of click entropy.

is that there is no explicit correspondence between these two. For our proposed approach LR_CE, we first automatically determine the threshold of click entropy for a query and then use Algorithm 1 to remove the ads. Our approach does not suffer from the disadvantages of the above two baselines.

Experiment Setting. All the experiments are carried out on a PC running Windows XP with AMD Athlon 64 X2 Processor(2GHz) and 2G RAM.

We use the predicted CTRs from the ad ranking task to approximate the term $P(ad|q)$ in Eq. 1 in this way: $P(ad|q) = \frac{CTR(ad)}{\sum_i CTR(ad_i)}$ where $CTR(ad)$ and $CTR(ad_i)$ are the predicted CTRs of the current ad and the i-th related ad for query q respectively. For the training, we use the feature "position"; while for testing, we set the feature "position" as zero for all instances.

5.2 Results and Analysis

#Removed ads vs. #Clicks. Figure 6(a) shows all the results of two baselines and our approach. From that, the main observations are as follows:

- **Performance.** The method LR_CTR obtains the optimal solution by the measure #click. Our approach LR_CE is near the optimal solution, and the baseline LR_RANDOM is the worst.
- **User specification.** From the viewpoint of the search engines, they may want to cut down a specific number of ads to reduce the number of irrelevant impressions while preserving the relevant ones. For addressing this issue, our approach LR_CE can first automatically determine the threshold of click entropy via the relationship in Figure 3 and then determine the displayed ads. This case cannot be dealt with by LR_CTR, because it needs to know all the click-through information in advance and then make a global analysis for removing irrelevant ads. Further, for a specific query, it can not determine exactly which ads should be displayed.

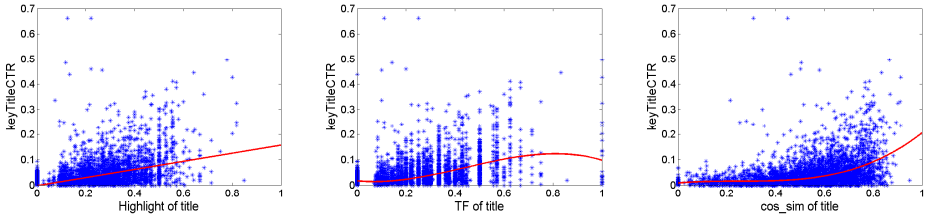
#Removed ads vs. CTR and #Clicks. Figure 6(b) shows how the total number of clicks and the number of removed ads vary with the threshold of click

entropy. As the threshold of click entropy increases, the total number of clicks increases while the number of removed ads decreases.

5.3 Feature Contribution Analysis

All the following analyses are conducted on the dataset DS_ExactMatch.

Features vs. keyTitleCTR. Figure 7 shows some statistics of the ad click-through data. When the values of features in 7 (a) and (c) increase, the keyTitleCTRs also increase, while the feature in 7 (b) increases, the keyTitleCTR first increases and then decreases.



(a)Highlight (b)TF (c)Cosine

Fig. 7. How keyTitleCTR varies with three different features.

Feature Ranking. Recursive feature elimination(RFE) uses greedy strategy for feature selection [22]. At each step, the algorithm tries to find the most useless feature and eliminate it. In this analysis, we use the measure Akaike Information Criterion (AIC) to select useful features. After excluding one feature, the lower the increase of AIC is, the more useless the removed feature is. The process will be repeated until only one feature left. Finally, we can get a ranking list of our features, and the top three are keyTitleCTR, position and cos_sim of title .

6 Related work

CTR-based advertisement. In this category, people try to predict CTRs, by which the query-related ads can be ranked. These methods can be divided into two main categories: click model [12, 23] and regression model [15].

Regarding click model, Agarwal et al. propose a spatio-temporal model to estimate CTR by a dynamic Gamma-Poisson model [1]. Craswell et al. propose four simple hypotheses for explaining the position bias, and find that the cascade model is the best one [9]. Chapelle and Zhang propose a dynamic Bayesian network to provide an unbiased estimation of relevance from the log data [5]. Guo et al. propose the click chain model based on Bayesian modeling[13].

Regarding regression model, Richardson et al. propose a positional model and leverage logistic regression to predict the CTR for new ads [17]. Chen et al. design and implement a highly scalable and efficient algorithm based on a linear

Poisson regression model for behavioral targeting in MapReduce framework [6].

There are also many other works [14]. For example, Dembczyński et al. propose a method based on decision rule [10].

Revenue-based advertisement. In this category, people try to take relevance or revenue into consideration rather than CTR while displaying ads.

Radlinski et al. propose a two-stage approach to select ads which are both relevant and profitable by rewriting queries [16]. Zhu et al. propose two novel learning-to-rank methods to maximize search engine revenue while preserving high quality of displayed ads [22]. Ciaramita et al. propose three online learning algorithms to maximize the number of clicks based on preference blocks [8]. Streeter et al. formalize the sponsored search problem as an assignment of items to positions which can be efficiently solved in the no-regret model [19]. Carterette and Jones try to predict document relevance from the click data [4].

Threshold-based methods. In this category, people try to utilize thresholds for determining whether to display ads or where to cut off the ranking list.

Broder et al. propose a method based on global threshold to determine whether to show ads for a query because showing irrelevant ads will annoy the user [3]. Shanahan et al. propose a parameter free threshold relaxation algorithm to ensure that support vector machine will have excellent precision and relatively high recall [18]. Arampatzis et al. propose a threshold optimization approach for determining where to cut off a ranking list based on score distribution [2].

7 Conclusion

In this paper, we study an interesting problem that how many ads should be displayed for a given query. There are two challenges: ad ranking and ad number prediction. First, we conduct extensive analyses on real click-through data of ads and the two main observations are 1) when the click entropy of a query exceeds a threshold the CTR of that query will be very near zero; 2) the threshold of click entropy can be automatically determined when the number of removed ads is given. Second, we propose a learning approach to rank the ads and to predict the number of displayed ads for a given query. Finally, the experimental results on a commercial search engine validate the effectiveness of our approach.

Learning to recommend ads in sponsored search presents a new and interesting research direction. One interesting issue is how to predict the user intention before recommending ads [7]. Another interesting issue is how to exploit click-through data in different domains where the click distributions may be different for refining ad ranking [21]. It would also be interesting to study how collective intelligence (social influence between users for sentiment opinions on an ad) can help improve the accuracy of ad number prediction [20].

Acknowledgments. Songcan Chen and Bo Wang are supported by NSFC (60773061), Key NSFC(61035003). Jie Tang is supported by NSFC(61073073, 60703059, 60973102), Chinese National Key Foundation Research (60933013, 61035004) and National High-tech R&D Program(2009AA01Z138).

References

1. D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *WWW'09*, pages 21–30, 2009.
2. A. Arampatzis, J. Kamps, and S. Robertson. Where to stop reading a ranked list?: threshold optimization using truncated score distributions. In *SIGIR'09*, pages 524–531, 2009.
3. A. Broder, M. Ciaramita, M. Fontoura, E. Gabrilovich, V. Josifovski, D. Metzler, V. Murdock, and V. Plachouras. To swing or not to swing: learning when (not) to advertise. In *CIKM'08*, pages 1003–1012, 2008.
4. B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS'07*, 2007.
5. O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW'09*, pages 1–10, 2009.
6. Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *KDD'09*, pages 209–218, 2009.
7. Z. Cheng, B. Gao, and T.-Y. Liu. Actively predicting diverse search intent from user browsing behaviors. In *WWW'10*, pages 221–230, 2010.
8. M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *WWW'08*, pages 227–236, 2008.
9. N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM'08*, pages 87–94, 2008.
10. K. Dembczyński, W. Kotłowski, and D. Weiss. Predicting ads' click-through rate with decision rules. In *TROA'08*, Beijing, China, 2008.
11. Z. Dou, R. Song, and J. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW'07*, pages 581–590, 2007.
12. G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR'08*, pages 331–338. ACM, 2008.
13. F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW'09*, pages 11–20, 2009.
14. M. Gupta. Predicting click through rate for job listings. In *WWW'09*, pages 1053–1054, 2009.
15. A. C. König, M. Gamon, and Q. Wu. Click-through prediction for news queries. In *SIGIR'09*, pages 347–354, 2009.
16. F. Radlinski, A. Z. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, and L. Riedel. Optimizing relevance and revenue in ad search: a query substitution approach. In *SIGIR'08*, pages 403–410, 2008.
17. M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW'07*, pages 521–530, 2007.
18. J. Shanahan and N. Roma. Boosting support vector machines for text classification through parameter-free threshold relaxation. In *CIKM'03*, pages 247–254, 2003.
19. M. Streeter, D. Golovin, and A. Krause. Online learning of assignments. In *NIPS'09*, pages 1794–1802. 2009.
20. J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD'09*, pages 807–816, 2009.
21. B. Wang, J. Tang, W. Fan, S. Chen, Z. Yang, and Y. Liu. Heterogeneous cross domain ranking in latent space. In *CIKM'09*, pages 987–996, 2009.
22. Y. Zhu, G. Wang, J. Yang, D. Wang, J. Yan, J. Hu, and Z. Chen. Optimizing search engine revenue in sponsored search. In *SIGIR'09*, pages 588–595, 2009.
23. Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *WSDM'10*, pages 321–330, 2010.