Online Learning from Click Data for Sponsored Search

Massimiliano Ciaramita, Vanessa Murdock, Vassilis Plachouras
Yahoo! Research, Ocata 1, Barcelona, 08003, Catalunya, Spain
{massi | vmurdock | vassilis}@yahoo-inc.com

ABSTRACT

Sponsored search is one of the enabling technologies for today's Web search engines. It corresponds to matching and showing ads related to the user query on the search engine results page. Users are likely to click on topically related ads and the advertisers pay only when a user clicks on their ad. Hence, it is important to be able to predict if an ad is likely to be clicked, and maximize the number of clicks. We investigate the sponsored search problem from a machine learning perspective with respect to three main sub-problems: how to use click data for training and evaluation, which learning framework is more suitable for the task, and which features are useful for existing models. We perform a large scale evaluation based on data from a commercial Web search engine. Results show that it is possible to learn and evaluate directly and exclusively on click data encoding pairwise preferences following simple and conservative assumptions. Furthermore, we find that online multilayer perceptron learning, based on a small set of features representing content similarity of different kinds, significantly outperforms an information retrieval baseline and other learning models, providing a suitable framework for the sponsored search task.

Categories and Subject Descriptors

H.3 [**INFORMATION STORAGE AND RETRIEVAL**]; H.3.5 [**Online Information Services**]: Commercial services; I.5 [**PATTERN RECOGNITION**]; I.5.1 [**Models**]: Neural Nets; I.5.2 [**Design Methodology**]: Classifier design and evaluation

General Terms

Algorithms, Design, Experimentation

Keywords

Sponsored search, ranking, online learning, perceptrons

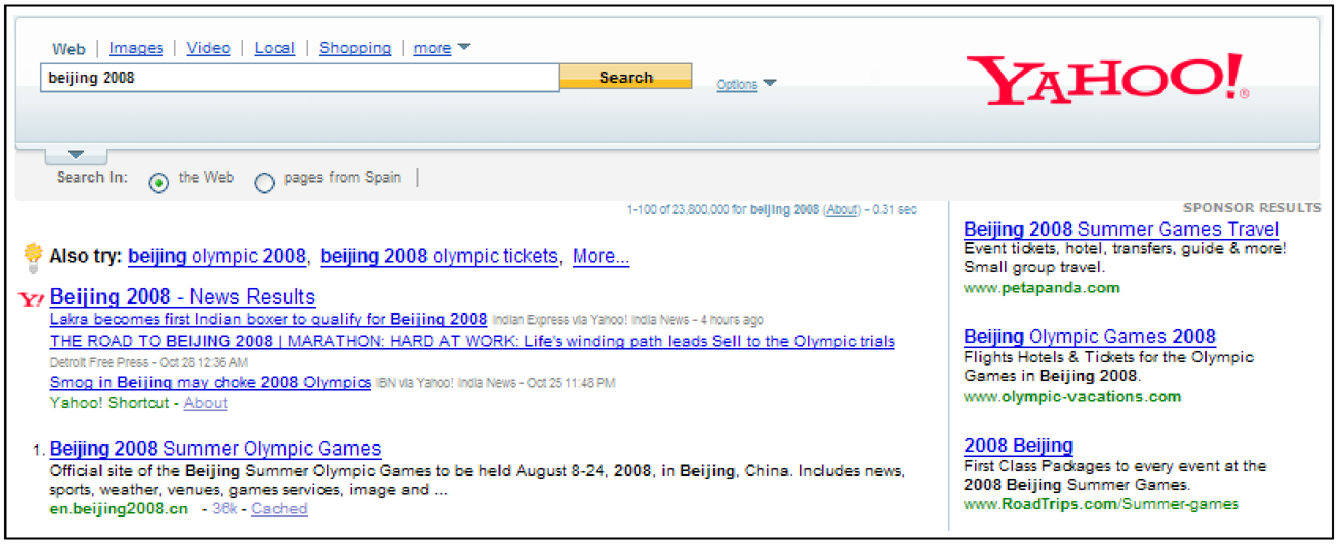
1. INTRODUCTION

Sponsored search is the task of placing ads that relate to the user's query on the same page as the search results returned by the search engine. Typically sponsored search results resemble search result snippets in that they have a title, and a small amount of additional text, as in Figure 1. When the user clicks on the title he is taken to the landing page of the advertiser. Search engine revenue is generated in large part by the sponsored results. Of the $16.9 billion generated by online advertising in 2006, 40% was generated by search advertising [12]. Clearly it is an advantage to the search engine to provide ads users will click.

While the final ranking takes into account the amount bidded by the advertiser and, more generally, the micro-economic model adopted by the search engine to maximize revenue, relevance, or quality of the ads returned is a crucial factor. Users are more likely to click ads that are relevant to their query. In fact, there is evidence that the level of congruence between the ad and its context has a significant effect even in the absence of a conscious response such as a click [34]. If we assume that congruency equates to topical similarity or "relevance", the task is to show the ads that are most similar to the user's query in the sponsored results. With this in mind we would like to place ads in the sponsored results that are a good match for the user's query. In this paper we abstract from the economic aspects of the problem and focus on the issue of improving the quality of the ads proposed as relevant to the query. Information about the quality of match at the content level can be taken into account later, including the micro-economic model, to compile the final list of ads to be displayed. While the publisher goal is maximizing profit, we stress the fact that maximizing ads quality or relevance is a crucial factor in this process.

Aside from the difficulties in assessing the similarity of an ad to a query that stem from the sparseness of the representation of both the query and the ad, the task of placing ads is complicated by the fact that users click on ads for a wide variety of reasons that are not reflected in the similarity of an ad to a query. For example, there is a strong positional bias to user clicks. Users are much more likely to click on items at the top of a ranked list of search results than items lower in the ranking [11, 14]. This makes using the click data to learn a ranking function over the ads and to evaluate the system more difficult. Specifically, user clicks are not an indication of absolute relevance. Rather, as Joachims proposed [13], the user click mainly indicates the clicked item is more relevant than the those items which were ranked higher but were not clicked. This observation implies that positive and negative examples can be extracted from query logs in a meaningful way, avoiding the complexities and noise of click-through rate estimation. Building on Joachims' suggestion we create a training set from the query logs of a real sponsored search system.

We propose that this type of data can be also used directly


Web | Images | Video | Local | Shopping | more
beijing 2008
Search
Options
YAHOO!
Search In: the Web pages from Spain
SPONSOR RESULTS

1-100 of 23,800,000 for beijing 2008 (About) - 0.31 sec

Also try: beijing olympic 2008, beijing 2008 olympic tickets, More...

Beijing 2008 - News Results
Lakra becomes first Indian boxer to qualify for Beijing 2008 Indian Express via Yahoo! India News - 4 hours ago
THE ROAD TO BEIJING 2008 | MARATHON: HARD AT WORK: Life's winding path leads Sell to the Olympic trials
Detroit Free Press - Oct 28 12:36 AM
Smog in Beijing may choke 2008 Olympics IBN via Yahoo! India News - Oct 25 11:48 PM
Yahoo! Shortcut - About

1. Beijing 2008 Summer Olympic Games
Official site of the Beijing Summer Olympic Games to be held August 8-24, 2008, in Beijing, China. Includes news, sports, weather, venues, games services, image and ...
en.beijing2008.cn - 36k - Cached

Beijing 2008 Summer Games Travel
Event tickets, hotel, transfers, guide & more! Small group travel.
www.petapanda.com

Beijing Olympic Games 2008
Flights Hotels & Tickets for the Olympic Games in Beijing 2008.
www.olympic-vacations.com

2008 Beijing
First Class Packages to every event at the 2008 Beijing Summer Games.
www.RoadTrips.com/Summer-games

Figure 1. Example of sponsored search advertisements on the search engine results page of Yahoo! for the query "Beijing 2008". Ads are shown on the right side under the label "SPONSOR RESULTS".

for evaluating a learning model. A few studies have already shown that click feedback from users can be used to improve machine learned ranking [32, 1]. However, this type of information has not been used for both training and evaluation. Previous work [32, 1] has ultimately relied on editorial data for evaluation. We show that consistent results are obtained in this way across different ranking methods and different feature sets.

We formulate the problem of ranking a set of ads given a query as a learning task and investigate three learning methods of increasing complexity based on the perceptron algorithm: a binary linear classifier, a linear ranking model and a multilayer perceptron, or artificial neural net. Specifically, we focus on online learning methods which suit the task of learning from large amounts of data, or from a stream of incoming feedback from query logs. As hoped, we find that accuracy increases with the complexity of the model.

Retrieving ads is complex in part because the text is sparse. In addition, features based on link structure that have been shown to benefit retrieval in a Web setting cannot be applied to ads because they lack an obvious link organization. Exploiting our setup we investigate several classes of features which have been proposed recently for content match, the task of ranking ads with respect to the context of a Web page, rather than a query. We start from the cosine similarity between query and ad. We decompose the ad and use as features the similarity of individual components of the ad and the query, and also properties of the degree of overlapping between the query and the ad [25]. Next we investigate a class of language-independent, knowledge free, features based on the distributional similarity of pairs of words which have been used successfully in content match [6], and can be extracted from any text collection or query log. These features measure the similarity between two texts independently of exact matches at the string level and are meant to capture indirect semantic associations. In content match there are many words that can be extracted from a Web page to compute such features, while in sponsored search there are only the terms in the query. We show that across all learning methods these features produce the best results. Overall, with a compact set of just twelve features we improve significantly over all traditional models.

This paper presents the following primary contributions. First, we show that click-data can be used directly for evaluation purposes which is a desirable property in the context of large scale systems that otherwise have to rely exclusively on editorial data, or carry out noisy estimations of click-through rates. Second, we show empirically that different methods of increasing complexity can be applied to the task and generate consistent results. This is important because it supports the hypothesis that the evaluation is consistent across different methods. On the learning side, it also shows that taking into account pairwise information in training is beneficial in machine-learned ranking, even in noisy settings. Finally, we provide empirical evidence on the utility of a class of simple features for ranking ads based on lexical similarity measures, which has possible applications to other query-based ranking tasks such as document retrieval and search in general.

The rest of the paper is organized as follows. In Section 2 we introduce the method for unbiasing click logs. Section 3 introduces several approaches to learning from click data for sponsored search. Section 4 describes the features evaluated in our experiments, which are discussed in Sections 5 and 6. Section 7 provides an overview of related work.

2. CLICK DATA

Employing user clicks to train and to evaluate a sponsored search system is a natural choice, since the goal in sponsored search is maximizing the number of clicks. However, user clicks cannot be used in a straight-forward manner because they have a strong positional bias [11, 14], and they only provide a relative indication of relevance [13]. There is a strong positional bias as highly ranked results or ads may be clicked because of their rank and not their relevance. For example, a user may click on the top ranked ad and then click on the third ad in the ranking, even if the third ad may be more relevant to his query. The reason for this bias is that users are likely to sequentially scan the ranked list of

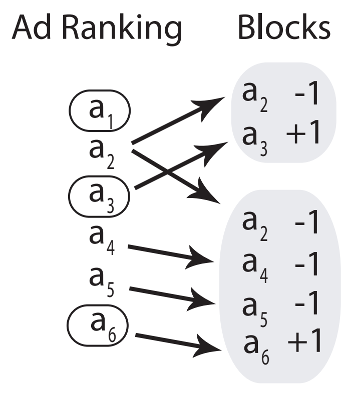


Figure 2. Illustrative example of how blocks are generated from the clicked and non-clicked ads for a query.

items and may click on an item before scanning the whole list, or may never scan the whole list at all.

To investigate how to employ user clicks to train and evaluate a sponsored search system, we have collected a set of queries and the ads that had been shown with them on the right-hand side of the search engine results page, from the logs of the Yahoo! Web search engine. We sampled queries until we collected a sufficiently large number of distinct clicked ads. We sampled only queries with three or more query terms because longer queries are more likely to lead to higher conversion rates [24]. In other words, users issuing longer queries are more likely to visit a Web site and perform a transaction. We also considered only one click for a query-ad pair from one user per day.

Following Joachims [13], we make a conservative assumption that a click can only serve as an indication that an ad is more relevant than the ads ranked higher but not clicked, but not as an absolute indication of the ad relevance. In this setting, the clicks on the top ranked ad do not carry any information, because the top ranked ad cannot be ranked any higher. In other words, there is no discriminative pairwise information associated with a click at position one. Hence, we do not consider such clicks in the remainder of our experiments. For each clicked ad, we create a *block* which consists of the clicked ad and the non-clicked ads that ranked higher, for a total of 123,798 blocks. In each block, we assign a score of "+1" to the clicked ad and "-1" to the ads that were ranked higher but were not clicked.

Figure 2 shows an example of the score assignment process. On the left-hand side of the figure, we show the ranking of six ads for a query. The ellipses around ads a_1, a_3 and a_5 denote that these ads were clicked by the user who submitted the query. The "gold-standard" blocks of ads, shown on the right-hand side of the figure, are generated in the following way. First we ignore the click on ad a_1 since this ad was already ranked first and it was clicked. Then, we form one block of ads with a_2 and a_3, assigning scores of "-1" and "+1", respectively. Next, we form a second block of ads consisting of a_2, a_4, a_5 with scores "-1" and a_6 with score "+1". In Figure 2 blocks of ads are shown with shaded areas.

From the example above we obtain two gold standard blocks from one user session. Alternatively, one could generate a single vector of binary values including all the positive and negative examples. One potential problem is that in such a way one introduces effectively an equivalence of relevance between clicked ads (positive points), while according to our conservative assumption we cannot really say anything about the relative relevance of two clicks in different blocks, because we do not have reliable means of interpreting clicks in an absolute way. As a matter of fact, in some of the ranking models we propose (classification and multilayer perceptron) the block structure is effectively ignored in training. However training on local pairwise preferences can be beneficial, as we show in the experimental section going from a global binary classification model to a ranking model which exploits the block structure. The block-based encoding is compatible with both approaches and defines a robust and intuitive learning task.

3. LEARNING MODELS

Learning with clicks can involve arbitrarily large amounts of data, or even learning from a continuous stream of data. *Online* learning algorithms are the most natural choice for this type of task, since the data need not be considered (or stored in memory) all at once; each pattern is used for learning in isolation. Notice how learning from blocks fits perfectly online learning in that any feedback can be immediately used to update the current model, even when a query or ad is observed for the first time, without the need to accumulate evidence for reliable estimate of click-through rates. As a general online learning framework we choose the perceptron algorithm. The perceptron was invented by Frank Rosemblatt in 1958 [26], and was initially criticized because of its inability to solve non-linear problems [22]. In fact, the perceptron, like support vector machines (SVM) and other methods, can learn non-linear models by means of kernel functions in dual algorithms, or by means of higher-order feature mappings in the primal form, or again (see below Section 3.3) by means of multilayer architectures.

The perceptron has received much attention in recent years for its simplicity and flexibility. Among other fields, the perceptron is popular in natural language processing, where it has been successfully applied to several tasks such as syntactic parsing, tagging, information extraction and re-ranking [7, 29, 30, 31]. We preferred the perceptron over other popular methods, such as SVM, for which incremental formulations have been proposed [5, 17], because accurately-designed perceptrons (i.e., including regularization, margin functions, etc.) often performs as well as more complex methods at a smaller computational cost. Moreover, the simplicity of the algorithm allows easy customization, which is crucial in large scale settings. In Section 3.4, we benchmarked one of our perceptron models on a ranking task, yielding results comparable to SVM and Boosting.

We investigate three approaches to learning to rank ads based on click data: classification, ranking, and non-linear regression. The general setting involves the following elements. A pattern $\mathbf{x} \in \mathbb{R}^d$ is a vector of features extracted from an ad-query pair (a, q). Each pattern $\mathbf{x}_i$ is associated with a response value $y_i \in \{-1, +1\}$. In classification we associate a vector for a pair which has not been clicked with -1, also referred to as class y_0, and a vector for a pair which has been clicked with +1, also referred to as class y_1. The goal of learning is to find a set of parameters, or weights, α used to assign a score $F(\mathbf{x}_i; \alpha)$ to patterns such that $F(\mathbf{x}_i; \alpha)$ is close to the actual value y_i. In particular, we are interested in predicting the clicked ad in a block of ads.

3.1 Classification

In a classification framework the goal is to learn a function which is able to accurately assign a pattern to either the clicked or non-clicked class. Patterns in the data are used independently of one another in training and the classifier simply finds a weight vector which assigns each pattern to the correct class[1]. After training, the classifier can be used to rank ads given a query; e.g., to identify the most likely clickable pattern in a block.

The basic classifier is a binary perceptron. We extend the basic model by *averaging* and adding an *uneven margin function.* Averaging is a method for regularizing the classifier by using – for prediction after training – the average weight vector of all perceptron models posited during training [10]. The uneven margin function is a method for learning a classifier with large margins for cases in which the distribution of classes is unbalanced [20]. Since non-clicked ads are more numerous than clicked ads we face an unbalanced learning task. The uneven margin function pushes learning towards achieving a larger margin on the positive class. The binary perceptron uses the *sign* function as a discriminant:

$$F(\mathbf{x};\alpha) = \mathrm{Sgn}(\langle \mathbf{x}, \alpha \rangle) \tag{1}$$

We learn α from the training data. The model has two adjustable parameters. The first is the number of instances T to use in training, or the number of passes (epochs) over the training data. The second concerns the uneven margin function, for which we define a constant τ_1 used in training to define a margin on the positive class. While training, an error is made on a positive instance $\mathbf{x}$ if $F(\mathbf{x};\alpha) \leq \tau_1$; the parameter on the negative class $\tau_0 = 0$ and is effectively ignored. The learning rule is:

$$\alpha^{t+1} = \alpha^t + y_i \mathbf{x}_i \tag{2}$$

The ranking function defined on the binary classifier is simply the inner product between the pattern and the weight vector:

$$\mathrm{S}_{\mathrm{apm}} = \langle \mathbf{x}, \alpha \rangle \tag{3}$$

In evaluation, we use this score to rank ads in each block.

3.2 Ranking

Another way of modeling click feedback is directly as a ranking problem. For this purpose we implement the ranking perceptron proposed by Shen and Joshi [30], which reduces the ranking problem to a binary classification problem. The general objective is to exploit the pairwise preferences induced from the data by training on pairs of patterns, rather than independently on each pattern. Let R_b be a set of pairs of patterns for a block b, such that $(\mathbf{x}_i, \mathbf{x}_j) \in R_b \iff r(y_i) < r(y_j)$, where $r(y_i)$ is the rank of $\mathbf{x}_i$ in b; i.e., in our case, either $y_i = 1$ and $r(y_i) = 1$, or $y_i = -1$ and $r(y_i) = 2$.

Given a weight vector α, the score for a pattern $\mathbf{x}$ is again the inner product between the pattern and the weight vector:

$$S_{rnk} = \langle \mathbf{x}, \alpha \rangle \tag{4}$$

However, the error function depends on pairwise scores. In training, for each pair $(\mathbf{x}_i, \mathbf{x}_j) \in R_b$, the score $S_{rnk}(\mathbf{x}_i - \mathbf{x}_j)$ is computed. Given a margin function g and a positive learning margin τ, if $S_{rnk}(\mathbf{x}_i - \mathbf{x}_j) \leq g(r(y_i), r(y_j))\tau$, an update is made as follows:

$$\alpha^{t+1} = \alpha^t + (\mathbf{x}_i - \mathbf{x}_j) g(r(y_i), r(y_j))\tau \tag{5}$$

In particular, because the discriminant function is an inner product, $S_{rnk}(\mathbf{x}_i - \mathbf{x}_j) = S_{rnk}(\mathbf{x}_i) - S_{rnk}(\mathbf{x}_j)$. By default we use $g(i,j) = (\frac{1}{i} - \frac{1}{j})$ as a margin function as suggested in [30]. Although there are only two possible ranks in our setting, the hope is that training on pairs provides more information than training on patterns in isolation. For regularization purposes, averaging is applied also to the ranking perceptron.

[1]Under the assumption that the training data is separable, i.e., there exists an hyperplane which correctly classifies all data points.

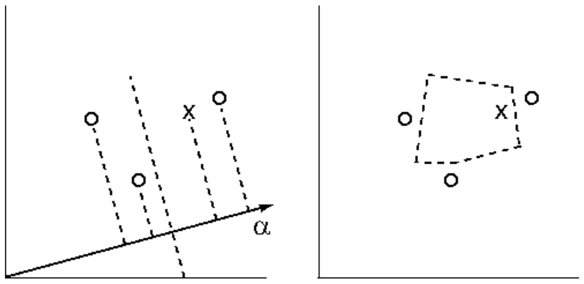


Figure 3. Examples of decision boundaries that can be learned by the linear, on the left, and non-linear models, on the right, in a two-dimensional case. X denotes positive (clicked) patterns while circles denote negative (non-clicked) patterns. The linear classifier depicted on the left correctly classifies the positive examples, but misclassifies a negative one. The non-linear classifier can find complex decision boundaries to solve such non-linearly separable cases.

3.3 Multilayer Regression

One possible drawback of the previous methods is that they are limited to learning linear solutions. To improve the expressive power of our ranking functions, within the online perceptron approach, we experimented also with multilayer models. The topology of multilayer perceptrons includes at least one non-linear activation layer between the input and the output layers. Multi-layer networks with sigmoid non-linear layers can generate arbitrarily complex contiguous decision boundaries [8], as shown in Figure 3, and have been used successfully in several tasks, including learning to rank [3]. The multilayer perceptron is a fully connected three-layer network with the following structure:

1. Input layer: d units $x_1, x_2, .., x_d$ + a constant input $x_0 = 1$
2. Hidden layer: n_H units $w_1, w_2, .., w_{n_H}$ + a constant weight $w_0 = 1$
3. Output layer: one unit z
4. Weight vector: $\alpha^2 \in \mathbb{R}^{n_H}$ + a bias unit α_0^2
5. Weight matrix: $\alpha^1 \in \mathbb{R}^{d \times n_H}$ + a bias vector $\alpha_0^1 \in \mathbb{R}^{n_H}$

The score $S_{mlp}(\mathbf{x})$ of a pattern $\mathbf{x}$ is computed with a feed-forward pass:

$$S_{mlp}(\mathbf{x}) = z = \sum_{j=1}^{n_H} \alpha_j^2 w_j + \alpha_0^2 = \langle \alpha^2, \mathbf{w} \rangle \tag{6}$$

where $w_j = f(net_j)$, and

$$net_j = \sum_{i=1}^{d} \alpha^1_{ij} x_i + \alpha^1_0 = \langle \alpha^1_j, \mathbf{x} \rangle \tag{7}$$

The activation function $f(.)$ of the hidden unit is the sigmoid:

$$f(net) = \frac{1}{1 + \exp^{-a\ net}}. \tag{8}$$

Supervised training begins with an untrained network with randomly initialized parameters. Training is carried out with backpropagation [27]. An input pattern $\mathbf{x}_i$ is selected, its score computed with a feedforward pass and compared to the true value y_i. Then the parameters are adjusted to bring the score closer to the actual value of the input pattern. The error $\mathbf{E}$ on a pattern $\mathbf{x}_i$ is the squared difference between the guessed score $S_{mlp}(\mathbf{x}_i)$ and the actual value y_i of $\mathbf{x}_i$, or for brevity $(y_i - s_i)$; thus $\mathbf{E} = \frac{1}{2}(y_i - s_i)^2$. After each iteration t, α is updated component-wise to α^{t+1} by taking a step in weight space which lowers the error function:

$$\begin{aligned} \alpha^{t+1} &= \alpha^t + \triangle\alpha^t \\ &= \alpha^t - \eta \frac{\partial \mathbf{E}}{\partial \alpha^t} \end{aligned} \tag{9}$$

where η is the *learning rate*, which affects the magnitude, or speed, of the changes in weight space. The weight update for the hidden-to-output weights is:

$$\triangle\alpha^2_i = -\eta \delta w_i \tag{10}$$

where $\delta = (y_i - z_i)$. The learning rule for the input-to-hidden weights is:

$$\triangle\alpha^1_{ij} = -\eta x_j f'(net_j) \alpha^1_{ij} \delta. \tag{11}$$

where f' is the derivative of the non-linear activation function.

3.4 Benchmark on a public dataset

A full comparison of the methods presented here with other methods on information retrieval tasks is beyond the scope of the paper. However, to get an estimate for the accuracy of the methods we implemented, we evaluated the ranking perceptron (see Section 3), on the Letor dataset [21]. On all evaluation metrics the ranking perceptron achieves scores comparable to SVM on the OHSUMED and TD2003 datasets, and comparable to RankBoost on TD2004. Thus in line with the best method in each dataset. The multilayer perceptron outperforms the ranking perceptron on exploratory runs, but we did not carry out extensive comparisons in this context.

4. FEATURES

A range of features, from simple word overlap and textual similarity features to statistical association between terms from the query and the ads, are used for learning a ranking model. In particular, we are interested in finding features which helps in the face of sparse data, as ads are characterized by small amounts of text.

4.1 Word Overlap

We compute four features that assess the degree of overlap between the query and the ad materials. The first feature has a value of one if all of the query terms are present in the ad:

$$\text{if } (\forall t \in q) t \in a, F_1 = 1, \text{ and 0 otherwise.} \tag{12}$$

The second feature has a value of one if some of the query terms are present in the ad:

$$\text{if } \exists t \in q \text{ such that } t \in a, F_2 = 1, \text{ and 0 otherwise.} \tag{13}$$

The third feature has a value of one if none of the query terms are present in the ad:

$$\text{if } \neg\exists t \in q \text{ such that } t \in a, F_3 = 1, \text{ and 0 otherwise.} \tag{14}$$

The fourth feature is the percentage of the query terms that have an exact match in the ad materials.

Prior to computing the features, both the query and the ad were normalized for case. For the purpose of word overlap, we chose to stem and stop less aggressively than we would do with functions that are smoothed. For this reason we used a Krovetz stemmer [18], and only single characters were removed.

4.2 Cosine similarity

We compute the cosine similarity $sim(q, a)$ between the query q and the ad a as follows:

$$sim(q, a) = \frac{\sum_{t \in q \cap a} w_{qt} w_{at}}{\sqrt{\sum_{t \in q} w^2_{qt}} \sqrt{\sum_{t \in a} w^2_{at}}} \tag{15}$$

where the weight w_t of a term in q or a corresponds to the $tf - idf$ weight:

$$w_t = tf \cdot \log_2 \frac{N + 1}{n_t + 0.5} \tag{16}$$

In Equation (16), tf is the frequency of a term in q or in a. When considering queries q, tf is expected to be uniformly distributed with one being the most likely value, because terms are not likely to be repeated in queries. In addition, N corresponds to the total number of available ads and n_t corresponds to the number of ads in which term t occurs.

The $tf - idf$ weight w_{at} of term t in a is computed in the same way. We have also computed the cosine similarity between q and each of the fields of the ads, that is, the ad title a_t, the ad description a_d and its bidded terms a_b. In all cases, we have applied Porter's stemming algorithm and we have removed stop words.

Cosine similarity has been used effectively for ranking ads to place on Web pages in the setting of contextual advertising [25, 6]. A difference with our setting is that in the case of contextual advertising, the cosine similarity is computed between the Web page and ad. While there are more complex similarity functions that have been developed and applied for the case of computing the similarity between short snippets of text [28], we use cosine similarity because it is parameter free and inexpensive to compute.

4.3 Correlations

Queries and ads are both short snippets of text, which tend not to have a high vocabulary overlap. To address this issue, and exploit additional information from non-matching terms, we consider two features based on measuring the statistical association of terms from an external corpus.

Feature Name	Abbrev.	Description
		Word Overlap Features
NoKey	O	1 if no query term is present in the ad materials; 0 otherwise
SomeKey		1 if at least one query term is present in the ad materials; 0 otherwise
AllKey		1 if every query term is present in the ad materials; 0 otherwise
PercentKey		The number of query terms present in the ad materials divided by the number of query terms
		Cosine Similarity Features
Ad	B	The cosine similarity between the query and the ad materials (baseline)
Title	F	The cosine similarity between the query and the ad title
Description		The cosine similarity between the query and the ad description
Bidterm		The cosine similarity between the query and the bidded terms
		Correlation Features
AvePMI	P	The average pointwise mutual information between terms in the query and terms in the ad
MaxPMI		The maximum pointwise mutual information between terms in the query and terms in the ad
CSQ	C	Number of query-ad term pairs that have a χ^2 statistic in the top 5% of computed χ^2 values.

Table 1. Summary of features. The column "Abbrev." provides an abbreviated name for one or more features, as they will be used in the experiments.

4.3.1 Pointwise Mutual Information

One measure of association between terms is pointwise mutual information (PMI). We compute PMI between terms of a query q and the bidded terms of an ad a. PMI is based on co-occurrence information, which we obtain from a set of queries submitted to the Yahoo! search engine:

$$PMI(t_1, t_2) = \log_2 \frac{P(t_1, t_2)}{P(t_1)P(t_2)} \qquad (17)$$

where t_1 is a term from q, and t_2 is a bidded term from the ad a. $P(t)$ is the probability that term t appears in the query log, and $P(t_1, t_2)$ is the probability that terms t_1 and t_2 occur in the same query.

We form the pairs of t_1 and t_2 by extracting the query terms and the bidded terms of the ad. We only consider pairs of terms consisting of distinct terms with at least one letter. For each pair (q, a) we use two features: the average PMI and the maximum PMI, denoted by AvePMI and MaxPMI, respectively.

4.3.2 χ^2 Statistic

Another measure of association between terms is the χ^2 statistic, which is computed with respect to the occurrence in a query log of terms from a query, and the bidded terms of an ad:

$$\chi^2 = \frac{|L|(o_{11}o_{22} - o_{12}o_{21})^2}{(o_{11} + o_{12})(o_{11} + o_{21})(o_{12} + o_{22})(o_{21} + o_{22})} \qquad (18)$$

where $|L|$ is the number of queries in the query log, and o_{ij} are defined in Table 2. For example o_{11} stands for the number of queries in the log, which contain both terms t_1 and t_2. Similarly, o_{12} stands for the number of queries in the log, in which term t_2 occurs but term t_1 does not. We compute the χ^2 statistic for the same pairs of terms on which we compute the PMI features. Then, for each query-ad pair, we count the number of term pairs that have a χ^2 higher than 95% of all the computed χ^2 values.

An overview of the features we use is shown in Table 1. All feature values were normalized by column across the entire dataset with a $z-score$, in order to have zero mean and unit standard deviation, thus each feature x_i was standardized as:

$$z = \frac{x_i - \mu_i}{\sigma_i}. \qquad (19)$$

In addition we augmented each data vector with a bias feature which has a value of one for every example, and serves as a prior on the response variable.

Before continuing, it is important to observe that all the features we described in this section can be computed efficiently. For example, the word overlap and cosine similarity features can be computed online, when a user query is received. The correlation features can also be computed online efficiently by using a look-up table with word co-occurrence information for pairs of words.

	t_1	$\neg t_1$
t_2	o_{11}	o_{12}
$\neg t_2$	o_{21}	o_{22}

Table 2. Definition of o_{ij} for the calculation of the χ^2 statistic in Equation 18.

Part	Development size	Test size
1	1358	1445
2	1517	1369
3	1400	1488
4	1408	1514
5	1410	1329

Table 3. The number of blocks in each of the development and test partitions of the data.

5. EXPERIMENTAL SETUP

We split the dataset into one training set, 5 development sets and 5 test sets, so that all the blocks for a given query are in the same set. The exact number of blocks for each of the development and test sets is given in Table 3. The training set consists of 109,560 blocks.

A ranking algorithm produces a score for each query-ad pair. The ads are ranked according to this score. Because of the way the data is constructed to account for the relative position of clicks, each block has only one click associated

	Classification		Ranking		Regression	
Feature set	Prec at 1	MRR	Prec at 1	MRR	Prec at 1	MRR
B	0.322	0.582 ±0.306	0.333	0.590 ±0.307	0.328	0.585 ±0.307
BO	0.319	0.578⋆ ±0.306	0.352	0.602⋆ ±0.310	0.343	0.596⋆ ±0.309
BF	0.341	0.593⋆ ±0.309	0.347	0.597⋆ ±0.310	0.374	0.615⋆ ±0.314
BFO	0.357	0.605⋆ ±0.311	0.357	0.605⋆ ±0.311	0.371	0.614⋆ ±0.313
BFOP	0.357	0.604⋆ ±0.311	0.359	0.606⋆ ±0.311	0.374	0.617⋆ ±0.313
BFOC	0.351	0.601⋆† ±0.310	**0.364**	**0.610⋆† ±0.311**	0.381	0.619⋆† ±0.315
BFOCP	**0.360**	**0.606⋆ ±0.311**	0.363	0.609⋆ ±0.311	**0.388**	**0.624⋆† ±0.315**

Table 4. The results for classification, ranking and regression, computed over all trials. The best result is indicated in bold. Results that are statistically significant with respect to the baseline are indicated with a star. Results indicated with a dagger are statistically significant with respect to the features B + F + O. The results for precision at one were not tested for statistical significance.

	Classification		Ranking		Regression	
Feature set	Prec at 1	MRR	Prec at 1	MRR	Prec at 1	MRR
B	0.322 ±0.008	0.582 ±0.003	0.333 ±0.014	0.590 ±0.006	0.331 ±0.020	0.586 ±0.012
BO	0.339 ±0.020	0.591 ±0.012	0.352 ±0.010	0.602 ±0.005	0.351 ±0.016	0.600 ±0.011
BF	0.340 ±0.016	0.592 ±0.007	0.345 ±0.007	0.596 ±0.004	0.368 ±0.013	0.611 ±0.007
BFO	0.356 ±0.007	0.604 ±0.004	0.359 ±0.006	0.605 ±0.003	0.375 ±0.016	0.616 ±0.008
BFOP	**0.359 ±0.008**	**0.606 ±0.005**	0.361 ±0.010	0.607 ±0.007	0.372 ±0.015	0.614 ±0.008
BFOC	0.350 ±0.011	0.600 ±0.009	**0.365 ±0.007**	**0.611 ±0.003**	0.381 ±0.010	0.619 ±0.005
BFOCP	0.357 ±0.014	0.605 ±0.008	0.364 ±0.006	0.609 ±0.003	**0.387 ± 0.009**	**0.622 ±0.004**

Table 5. The average of five trials, for classification, ranking and regression. The standard deviation refers to the 5 trials, not the standard deviation within a given trial. The best result is shown in bold.

with it. For this reason we evaluate the precision at rank one, which tells us how many clicked ads were placed in the first position by the ranker, and the mean reciprocal rank, which tells us the average rank of the clicked ad in the output of the ranker. The mean reciprocal rank is computed as:

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{rank_i} \quad (20)$$

where $rank_i$ is the rank of the clicked ad according to the ranking function score and n is the total number of blocks. The MRR score gives an indication of how far on average the ranker's guess is in the ranked list, thus providing a smoother measure than precision at one.

5.1 Model selection

All adjustable parameters of the learning models were fixed on the development datasets. The best values were selected by monitoring the average accuracy over the 5 development folds, the optimal values on development were used on the evaluation set. We trained all models with a stochastic protocol, choosing a training instance at random without replacement: a block for the ranking case, a single pattern for the classification and multilayer models.

In the classification case we set the parameters T and τ. We tried three values for τ_1, 1, 10 and 100, and found 100 to give the best results. As for the number of iterations, we found that all models (not only in classification) tended to converge quickly, rarely requiring more than 20 iterations to find the best results.

In the ranking model, in addition to the number of iterations T we optimize the positive learning margin τ. We obtained the best results with the value $\tau = 1$, which we used in all experiments with ranking perceptron.

The multilayer model has a number of adjustable parameters, some of the parameters were kept with default values; e.g., the sigmoid parameter $a = 1.716$ [8]. Following [8] we initialized the network weights for the hidden-to-output units uniformly at random in the interval $-\frac{1}{\sqrt{(nH)}} < \alpha_i^2 < \frac{1}{\sqrt{(nH)}}$, the input-to-hidden weights were initialized randomly in the interval $-\frac{1}{\sqrt{(d)}} < \alpha_{ij}^2 < \frac{1}{\sqrt{(d)}}$. On the development data we found that hidden layers with 50 units, and $\eta = 0.01$, produced fast training and stable results, we kept these values fixed on all experiments involving the multilayer model. The number of iterations was set on the development set, running a maximum of 50 iterations[2].

6. RESULTS

The baseline model has only one feature, the cosine similarity between the ad and the query with $tf - idf$ weights. Table 4 shows the results for classification, ranking, and multilayer regression for each of the five test sets concatenated. That is, for the five test folds evaluated as one dataset, in order to compute the significance of the mean reciprocal rank results. For mean reciprocal rank we used a paired t-test. Results indicated with a star are significant at the $p < 0.05$ level with respect to the baseline. Most of the significant results are significant at the $p < 0.01$ level with respect to the baseline. The precision at one results were not tested for statistical significance. The significance for this metric is not computed because it is not appropriate for binary data.

We see that multilayer regression outperforms both classification and ranking, and further that the correlation features are a significant improvement over the other models. For one third of the examples in the evaluation, the predictor correctly identifies that the first result was clicked, and an MRR of 0.60 indicates that on average the clicked result was between rank one and rank two.

[2]One full iteration of the MLP model takes about 10 seconds on a desktop machine.

Although the standard deviation is high, around 0.3 for each model, this is to be expected because MRR can take values of 1, 0.5, 0.34, 0.25, 0.20, 0.18, etc. An average MRR of 0.6 indicates that most clicked ads were at positions 1, 2 and 3. If half of the results had an MRR of 1, and the other half had an MRR of 0.34, the mean would be 0.67, resulting in a standard deviation of 0.34. Thus the high standard deviation in this case is an artifact of the metric.

We also compute the averages and standard deviation across the five test sets, shown in Table 5. As indicated by the standard deviation for the trials, the method is robust to changes in the data set, even for precision at 1 which is in general a much less stable evaluation metric. As already shown for content match [25, 19, 6] weighting the similarity of each component separately and adding features about the degree of overlapping between query and ad improve significantly over the baseline. The best result for each model are achieved adding the term correlation features.

6.1 Discussion

Sponsored search click data is noisy, possibly more than search clicks. People, and fraudulent software, might click on ads for reasons that have nothing to do with topical similarity or relevance. While it is not obvious that we can learn to distinguish relevant from non-relevant ads based on a user click, we find that there is enough signal in the clicks that, with a simple method for unbiasing the rank of the click [13], it is possible to learn and carry out meaningful evaluation without the need for editorial judgments produced manually, or complex estimation of click-through rates. Arguably, evaluating a classifier on the task of identifying the ad which will be clicked is more directly related to the task of successfully ranking ads then guessing indirectly the relevance assigned by humans. However, since the two are strongly related it makes sense to investigate their interaction, in particular in learning. An interesting question is if, in the presence of both types of information, it might be beneficial to solve both tasks at the same time. This type of approach in fact fits well the artificial neural net learning methods [8] and would provide an interesting extension of the multilayer model.

The non-linear multilayer perceptron outperforms both linear models. Interestingly, both linear models when using also the correlation features seem to perform better when only use one (PMI or chi-squared) rather than both, (see Table 5). This might depend on the fact that the features are strongly correlated and the linear classifier does not posses enough information to prefer one over the other in case of disagreements. Thus it finds a better solution by trusting always one over the other. The non-linear model instead has enough expressive power to capture subtler interactions between features and achieves the best results using both features. Another interesting aspect is that, although there are only two possible rankings, and thus the problem boils down to a binary classification task, the linear ranking perceptron outperforms the simpler classifier. The difference seems to lie in the way training is performed, by considering pairwise of patterns. Previous work on learning to rank [3] has proposed methods for training on pairs of patterns in multilayer architectures. This is a natural direction for further investigation of this type of model.

In terms of the features, even the simple word overlap features produced statistically significant results over the baseline model. Since we are effectively re-ranking ad candidates retrieved by a retrieval system which we treat as a black box, candidates are biased by the initial ad placement algorithm, and it is possible that the initial retrieval system preferred ads with a high degree of lexical overlap with the query, and the word overlap features provided a filter for those ads. The correlation features, which capture related terms rather than matching terms, added a significant amount of discriminative information. Such features are particularly promising because they are effectively language-independent and knowledge free. Similar statistics can be extracted from many resources simple to compile, or even generated by a search engine. Overall these findings suggest both that relevant ads contain words related to the query, and that related terms can be captured efficiently with correlation measures such as pointwise mutual information and the chi-squared statistic. There are several opportunities for further investigation of this type of features, for example by machine translation modeling [23].

One limitation of the current way of modeling click data is that "relevance" judgments induced by the logs are strictly binary. We have seen that using pairwise information is useful in training and it may be desirable to generate more complex multi-valued feedback. Joachims *et al.* [14] proposed manipulating the presentation of candidates to users by swapping the order of contiguous candidates to obtain likelihood ratios. An interesting question is how such information might be automatically extracted from query logs.

7. RELATED WORK

Sponsored search can be thought of as a document retrieval problem, where the ads are the "documents" to be retrieved given a query. As a retrieval problem, sponsored search is difficult because ad materials contain very few terms. Because the language of the ads is so sparse, the vocabulary mismatch problem is even more critical. Jones *et al.* [15] approach the problem of vocabulary mismatch by generating multiple rewrites of queries to incorporate related terms. In their system, related terms are derived from user sessions in the query logs, where query rewrites have been identified. The set of possible rewrites is constrained to contain only terms that are found in the database of advertising keywords. They use a machine-learned ranking to determine the most relevant rewrite to match against the ads. In a follow on to this work, Zhang et al. [35] use active learning to select the examples to use in training machine-learned ranking. Both systems were evaluated on manual editorial judgments. By contrast our system uses click data both for training and evaluating the system. Furthermore, our models learn a ranking over the ads given a query directly, rather than learning a ranking over query rewrites.

Advertisements are represented in part by their keywords. In one model of Online advertising, ads are matched to queries based on the keywords, and advertisers bid for the right to use the keywords to represent their product. So a related task is keyword suggestion, which can be applied to sponsored search or to its sister technology, contextual advertising, which places an ad in a Web page based on the similarity between the ad and the Web page content. Carrasco *et al.* [4] approach the problem of keyword suggestion by clustering bi-partite advertiser-keyword graphs. Yih *et al.* [33] extract keywords from Web pages using features such as the term frequency, the inverse document frequency, and the presence of terms in the query logs. They stop short of using the keywords in an application.

Although we do not consider economic factors in our work, implicit in the task of ranking ads to improve the relevance at the top of the list, is the desire to increase user clicks. Feng et al. [9] investigate the effects of four different ranking mechanisms for advertisements in sponsored search on revenue. Two of the mechanisms depend wholly on the bid price of the terms. Of the remaining two, one proposes ranking ads by their expected clickthrough rate, and the other by a function of the bid price and the expected clickthrough rate. The focus of this work is on the economic impact of the advertising placement mechanism, and uses a simulated environment which treats as a variable the degree of correlation between the advertiser's willingness to pay for advertising slots, and the amount of traffic they attract. They find that ranking ads by a function of the bid price and the clickthrough rate outperforms other ranking mechanisms. They further explore using the clickthrough rate to dynamically update the ranking mechanism. They investigate two weighting schemes for user clicks. In the unweighted scheme, an ad's clickscore is increased by 1 every time it is clicked. In the weighted scheme the ad's clickscore is increased by δ^i where i is the rank of the clicked ad, which has the effect of damping the scores of clicks in the top positions, and boosting the scores of ads in lower positions, which is one way of addressing the positional bias in user clicks. There are several key differences between this work and ours. First, the evaluation metric in Feng *et al.* is the expected revenue, whereas we evaluate the relevance of the ads. The work by Feng *et al.* is done in a simulated environment, whereas ours is performed on data from the search engine. They use the clickthrough data as a measure of absolute relevance, rather than as a measure of relevance relative to other ads that were not clicked. Finally, and most importantly, they are assessing the intrinsic relevance of an ad, independent of the context of the user's query.

Contextual advertising is a sister technology to sponsored search, and many of the techniques used to place ads in Web pages may be used to place ads in response to a user's query. As with sponsored search, contextual advertising is usually a pay-per-click model, and the ad representations are similar in both sponsored search and contextual advertising. The primary difference is that rather than matching an ad to a query, the system matches the ad to a Web page.

Contextual advertising also suffers from the vocabulary mismatch problem. To compensate for this, Ribeiro-Neto *et al.* [25] augment the representation of the target page using additional Web pages. As a follow-on to this work, Lacerda *et al.* [19] select ranking functions using genetic programming, maximizing the average precision on the training data. Broder *et al.* [2] use a semantic taxonomy to find topically similar terms, to facilitate the match between an ad and a Web page.

The current paper is a follow-on to work in contextual advertising, presented in [6] and [23]. Key differences in the current work are the use of click data in place of human edited relevance judgments, both for learning a ranking function and for evaluation, the application to sponsored search rather than content match, and the use of several different types of classifiers.

Joachims [13] proposes that user clicks can be used to learn ranking functions for search engine results by considering that a user click is an indicator of relative relevance. That is, a click at rank j indicates that the document at rank j is more relevant than unclicked documents that preceded it in the ranked list. We adopt this ranking mechanism directly in our work, as described in Sections 2 and 5, however in addition to using it for training, we also use this ranking mechanism for evaluation. Also, Joachims [13] is the first in a series of work investigating using implicit user feedback, such as user clicks, for learning ranking functions. An overview of using implicit user feedback as surrogates for editorial relevance judgments can be found in [16].

8. CONCLUSION

In this paper we investigated an approach to learning and evaluating sponsored search ranking systems based exclusively on click-data and a simple conservative unbiasing method, which fits together well with online learning protocols. In particular, we focused on modeling the textual content which is a fundamental aspect of the ad selection problem. We found empirically that our approach produces consistent results across different learning models, of varying complexity, and across different feature representations. We found that learning on pairs of patterns is beneficial and that multilayer perceptrons provide a competitive platform for ranking from noisy data and compact feature representations. We also showed how simple and efficient semantic correlation features provide a valuable source of discriminative information in a complex task such as sponsored search, characterized by sparse textual descriptions. Interesting avenues for further research might involve other ranking tasks where objects have small textual descriptions, such as image and multimedia retrieval, more sophisticated multi-layer ranking functions trained on pairwise preferences, and other variants of the simple unbiasing method used here.

9. REFERENCES

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

[2] 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. ACM Press, 2007.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 89–96, 2005.

[4] J. Carrasco, D. Fain, K. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graph. In *Proceedings of the Workshop on Clustering Large Datasets, IEEE Conference on Data Mining.* IEEE Computer Society Press, 2003.

[5] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In

Advances in Neural Information Processing Systems, pages 409–415, 2000.

[6] M. Ciaramita, V. Murdock, and V. Plachouras. Semantic associations for contextual advertising. *International Journal of Electronic Commerce Research - Special Issue on Online Advertising and Sponsored Search*, 9(1), pages 1–15, 2008.

[7] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL 2004*, 2004.

[8] R. Duda, P. Hart, and D. Stork. *Pattern Classification (2nd ed.)*. Wiley-Interscience, 2000.

[9] J. Feng, H. Bhargava, and D. Pennock. Implementing sponsored search in web search engines: Computational evaluation of alternative mechanisms. *INFORMS Journal on Computing*, 19(1):137–148, 2007.

[10] Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[11] L. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in WWW search. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 478–479, 2004.

[12] IAB. Internet advertising revenue report: 2006 full-year results, 2007. http://www.iab.net/resources/.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2002.

[14] T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 154–161, 2005.

[15] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, 2006.

[16] D. Kelly. Implicit feedback: Using behavior to infer relevance. In *New Directions in Cognitive Information Retrieval*, pages 169 – 186. Springer Publishing, 2005.

[17] J. Kivinen, A. Smola, and R. Williamson. Online learning with kernels. In *Advances in Neural Information Processing Systems*, pages 785–792, 2001.

[18] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

[19] A. Lacerda, M. Cristo, M. Goncalves, 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. ACM Press, 2006.

[20] Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola. The perceptron algorithm with uneven margins. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 379–386, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[21] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of the SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[22] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, MA, 1969.

[23] V. Murdock, M. Ciaramita, and V. Plachouras. A noisy channel approach to contextual advertising. In *Proceedings of the 1st International Workshop on Data Mining and Audience Intelligence for Advertising (ADKDD'07)*, pages 21–27, 2007.

[24] OneUpWeb. How keyword length affects conversion rates, 2005. http://www.oneupweb.com/landing/keywordstudy_landing.htm.

[25] B. Ribeiro-Neto, M. Cristo, P. Golgher, and E. D. 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. ACM Press, 2005.

[26] F. Rosemblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[27] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representation by backpropagating errors. *Nature*, 323(99):533–536, 1986.

[28] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, pages 377–386, New York, NY, USA, 2006. ACM.

[29] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology and North-American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2003.

[30] L. Shen and A. Joshi. Ranking and reranking with perceptron. *Machine Learning. Special Issue on Learning in Speech and Language Technologies*, 60(1-3):73–96, 2005.

[31] M. Surdeanu and M. Ciaramita. Robust information extraction with perceptrons. In *Proceedings of NIST Automatic Content Extraction Workshop (ACE)*, 2007.

[32] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM)*, pages 118–126, New York, NY, USA, 2004. ACM.

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

[34] C. Y. Yoo. *Preattentive Processing of Web Advertising.* PhD thesis, University of Texas at Austin, 2006.

[35] W. V. Zhang, X. He, B. Rey, and R. Jones. Query rewriting using active learning for sponsored search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.