The Anatomy of an Ad: Structured Indexing and Retrieval for Sponsored Search

Michael Bendersky[†], Evgeniy Gabrilovich[‡], Vanja Josifovski[‡], and Donald Metzler[‡]

[†] Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003
[‡] Yahoo! Research, 4301 Great America Parkway, Santa Clara, CA 95054
bemike@cs.umass.edu | {gabr | vanjaj | metzler}@yahoo-inc.com

ABSTRACT

The core task of sponsored search is to retrieve relevant ads for the user's query. Ads can be retrieved either by exact match, when their bid term is identical to the query, or by advanced match, which indexes ads as documents and is similar to standard information retrieval (IR). Recently, there has been a great deal of research into developing advanced match ranking algorithms. However, no previous research has addressed the ad indexing problem. Unlike most traditional search problems, the ad corpus is defined hierarchically in terms of advertiser accounts, campaigns, and ad groups, which further consist of creatives and bid terms. This hierarchical structure makes indexing highly non-trivial, as naïvely indexing all possible "displayable" ads leads to a prohibitively large and ineffective index. We show that ad retrieval using such an index is not only slow, but its precision is suboptimal as well. We investigate various strategies for compact, hierarchy-aware indexing of sponsored search ads through adaptation of standard IR indexing techniques. We also propose a new ad retrieval method that yields more relevant ads by exploiting the structured nature of the ad corpus. Experiments carried out over a large ad test collection from a commercial search engine show that our proposed methods are highly effective and efficient compared to more standard indexing and retrieval approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Structured retrieval, sponsored search

1. INTRODUCTION

Textual ads are ubiquitous on the Web today, as they appear on a wide variety of Web pages ranging from obscure forums to major search engine result pages. Online textual ads connect advertisers to prospective customers, and clicks on these ads by Web users account for a significant fraction of the revenue for many online businesses: from news publishers to blogs to Web search engines. As of 2008, textual advertising comprised approximately 40% of the $22 billion online advertising market, which is predicted to double in size over the next 5 years.[1].

Textual ads are distributed via two primary channels, namely, *sponsored search* and *content match*. In sponsored search, textual ads are shown alongside of the Web search results, while in content match, contextually relevant ads are displayed on third-party Web pages. Historically, sponsored search evolved by allowing advertisers to explicitly bid on queries (bid terms[2]) that they wished to display their ads for. In this paradigm, the burden of ad selection was placed primarily on the advertisers, since they needed to skillfully choose a comprehensive list of queries relevant for their ads. This scenario is called *exact match*, since the query and the bid term must match exactly for an ad to be shown.

However, it quickly became apparent that it is virtually impossible to explicitly enumerate billions of less popular "tail" queries, hence it is impractical to cover them via exact match, even though such queries provide valuable advertising opportunities. To unlock the revenue potential of these numerous yet individually infrequent queries, the *advanced match* method was introduced. Here, the query and the bid term no longer need to match exactly, and ads are selected algorithmically by the search engine.

Recently, information retrieval techniques have been proposed for advanced match by indexing the ads as documents using the ad text visible to the user as well as the ad's bid terms [25, 4, 5]. Ads are then selected using an *ad query* that is generated from the user's query (sponsored search) or the Web page on which ads are to be displayed (content match). The ad query is executed against the index of ads using standard IR matching and ranking techniques. Most implementations of advanced match make the simplifying assumption that ads are atomic units that are independent of each other, even though ads from the same advertiser could be quite similar or nearly identical.

In practice, however, textual ads are defined and organized as a structured database with several types of entities, as shown in Figure 1. Each advertiser has one or more *accounts*. Each account in turn contains several *ad campaigns*, which have different temporal or thematic goals (e.g., sale of home appliances during the holiday season). Campaigns

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

[1] Source: eMarketer.com

[2] *Bid terms* are also sometimes referred to as *bid phrases* or simply *terms* in a sponsored search setting. Therefore, we use these concepts interchangeably throughout the paper.

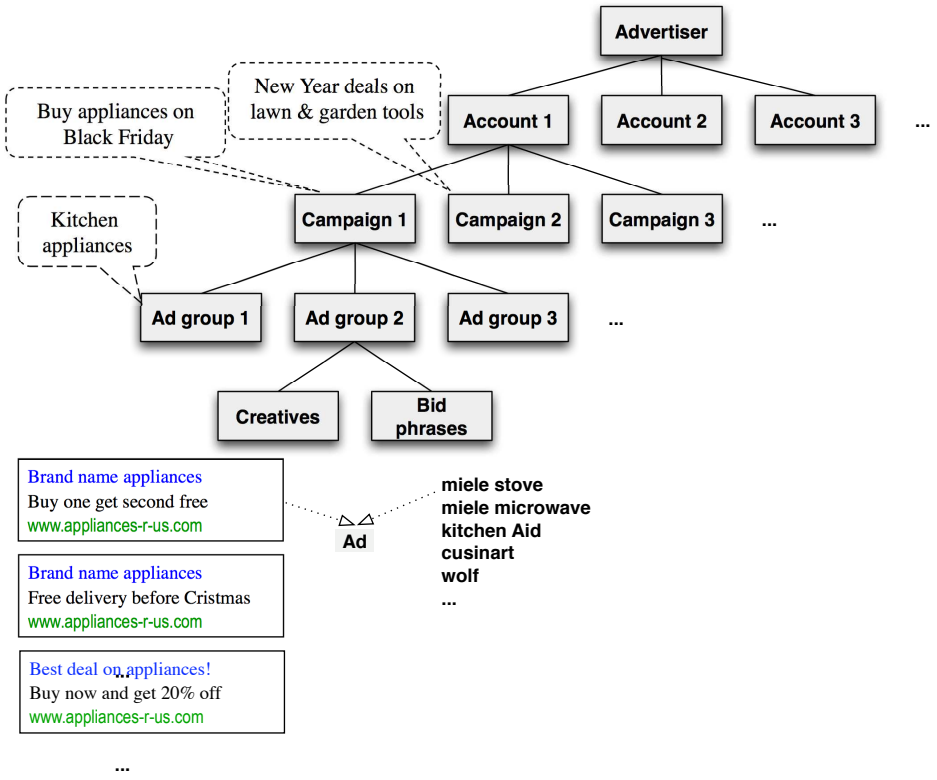


Figure 1: Schema of an ad database

consist of *ad groups*, which include multiple *creatives* (the visible text of the ad) and bid terms. Bid terms correspond to different products or services offered by the advertiser, while creatives represent different ways to advertise those products. Any creative can be paired with any bid term within the same ad group to create an actual ad displayed to the user. Search engines usually allow ad groups to contain a few dozen creatives and up to a thousand bid terms. This type of ad schema has been designed with the advertisers' needs in mind, as it allows the advertisers to easily define a large number of ads for a variety of products and marketing messages.

In view of this hierarchical definition, the ad retrieval problem can be formulated as the retrieval of $\langle creative, term \rangle$ pairs from a structured schema. In this paper we study the key issues in postulating ad retrieval as a *structured retrieval problem*, where the unit of retrieval is defined hierarchically. We also discuss several crucial tradeoffs that must be analyzed in this unique retrieval scenario.

Naïvely indexing all possible retrieval units (i.e., all possible $\langle creative, term \rangle$ combinations) using standard IR indexing approaches would result in a significant amount of wasted storage, since each creative and bid term will be indexed multiple times due to the Cartesian product semantics. Most of the inverted indexing algorithms used in modern search engines incur increased cost with larger index sizes, making the naïve approach infeasible in practice. To avoid this problem, we explore several hierarchical indexing schemes that significantly reduce the amount of duplication. In our study, we also quantify the impact that different indexing strategies have on ad retrieval effectiveness. We further develop a novel ranking algorithm that exploits the hierarchical structure and is both more efficient and more effective than the naïve indexing approach. This is done by employing a multi-phase retrieval approach, where we first retrieve an ad group, then select an optimal creative, and finally choose a bid term that makes the resulting ad the most relevant for the given query.

The main contributions of this paper are threefold. First, we propose novel, efficient ways of indexing sponsored search ads. We first transform the ad corpus to a collection of hierarchically structured textual documents, and then adapt standard IR indexing techniques to construct a compact yet effective ad index. The approaches we propose can also be applied to other retrieval tasks where the retrieval units are structured hierarchically. Second, we propose several ranking strategies that leverage the hierarchical structure of the ad corpus to achieve more accurate ad ranking. Finally, we conduct a large-scale evaluation using sponsored search data from a large commercial search engine that helps quantify, in a real world setting, the usefulness of our proposed structured indexing and retrieval approaches.

The remainder of the paper proceeds as following. In Section 2 we survey the related work. In Sections 3 and 4 we explain our novel structured indexing and retrieval approaches, respectively. Section 5 discusses our experimental evaluation carried out over real-life sponsored search data from a large commercial search engine. Finally, Section 6 concludes the paper.

2. BACKGROUND AND RELATED WORK

Textual advertising is a significant source of revenue for modern Web search engines. It is therefore not surprising that there is a growing research interest in designing better algorithms for sponsored search. Advanced match has been a focus of a few recent studies, most of which focus on the query rewriting paradigm [13, 23, 15]. However, recent work shows that employing a more flexible advanced match paradigm, which uses IR techniques to index ads as documents, leads to significant improvements in sponsored search performance [4, 25, 5]. Standard IR techniques such as *tf-idf* weighting [4], pseudo-relevance feedback [5], query reformulation [3, 25], clustering [29], document categorization [4, 1] and supervised learning of optimal ranking functions [16] have been shown to be highly beneficial for improving the quality of the retrieved ads.

Structured indexing and retrieval is a research field that has received increasing attention, especially due to the spreading adoption of the XML format for representing documents [10, 17]. Techniques that take into account document structure were shown to be beneficial in a variety of domains ranging from Web search [26] to book retrieval [14] to desktop search [9]. To the best of our knowledge, our work is the first to apply the principles of structured retrieval in sponsored search.

In many domains, a simple linear combination of scores from different document fields is sufficient for improving retrieval performance in the structured retrieval setting [2, 7, 20]. In the case of ad retrieval, however, such a straightforward approach is not sufficient. As we show in Section 4.1, due to the heterogeneity inherent in the ad databases, more elaborate feature design and parameter estimation techniques are necessary to achieve performance gains over the baseline that does not employ any structural information.

Structure was also recently found to be useful for the text classification task [22]. Specifically for sponsored search, Raghavan and Hillard [24] reported that a weighted combination of features based on creative and term fields helps reduce the number of irrelevant ads shown to the user (a task called ad filtration). In this work, we show that adding features that exploit the hierarchical structure of ad groups helps to further improve the accuracy of ad filtration.

3. AD INDEXING AND RETRIEVAL

As explained in Section 1, our target retrieval unit is a $\langle creative, term\rangle$ pair, which together comprise a displayable ad. Thus, our ultimate goal is to produce a ranked list of relevant $\langle creative, term\rangle$ pairs.

Our index contains two types of *fields*: *creative* fields (c) and *term* fields (t). Each creative field consists of three sub-fields: *title*, *description* and *URL*, as shown in Fig. 1. Each ad group g consists of several creative and term fields, grouped by an advertiser (see Fig. 1). The set of all ad groups is denoted $\mathbf{G}$, and the sets of all creatives and terms (for all the ads groups) are denoted $\mathbf{C}$ and $\mathbf{T}$, respectively. The set of creatives or terms associated with a specific ad group g is denoted $\mathbf{C}^g$ and $\mathbf{T}^g$, respectively. Hence, the total number of *unique* fields associated with a given ad corpus is:

$$|\mathbf{C}| + |\mathbf{T}| = |\mathbf{G}|(|\overline{\mathbf{C}^g}| + |\overline{\mathbf{T}^g}|), \quad (1)$$

where $|\overline{\mathbf{C}^g}|$ and $|\overline{\mathbf{T}^g}|$ denote the average number of creatives and terms per ad group, respectively.

3.1 Scoring Function and Ranking

Although our structured retrieval approach is general enough to incorporate any type of scoring functions, in this work we take the well known language modeling approach to information retrieval [8, 33]. In this approach, indexing units are scored by their probability of generating the query terms [21]. Formally, given a query q and an indexing unit u, we score each unit using a unigram language model

$$p(q|u) = \prod_{w_i \in q} p(w_i|u),$$

where $p(w_i|u)$ is estimated using Dirichlet smoothing [34], so that the final scoring function is as follows:

$$sc_q(u) \triangleq \sum_{w_i \in q} \log \frac{tf_{w_i,u} + \mu \frac{tf_{w_i,\mathcal{C}}}{\sum_{w_j \in \mathcal{C}} tf_{w_j,\mathcal{C}}}}{|u| + \mu}, \quad (2)$$

where $tf_{w,k}$ is the number of occurences of a term w, in either a particular indexing unit ($k = u$) or the entire collection ($k = \mathcal{C}$), and μ is a free parameter, which controls the amount of smoothing.

Indexing units are then ranked in descending order, based on their score. As the user typically only sees a limited number of sponsored search results in response to her query, we assume that only a list of top K indexing units $[u_{(1)}, \ldots, u_{(K)}]$ is retrieved in practice. Each unit u in the list is associated with an ad group identifier gId_u. To promote diversity and advertiser coverage in the ranked list, we can require that $[gId_{u(1)}, \ldots, gId_{u(K)}]$ are unique, thereby limiting the number of ads retrieved from a single ad group to one.

3.2 Structured Ad Indexes

We now describe three proposed indexing strategies for sponsored search. These strategies take into account the ad structure outlined in Section 1. The strategies presented in this paper investigate the indexing of the two lower levels of the hierarchal ad structure (Fig. 1), namely the ad group and the creative-bid term levels. The underlying principles of these strategies, however, are general enough to easily extend them to the higher levels of ad hierarchy, which we intend to do as part of future work.

The strategies presented differ in their choice of the atomic indexing unit and their ranking algorithms. Table 1 provides a formal definition of indexing units and estimated index sizes for each strategy. In what follows, we provide a detailed description of these indexing strategies and their applications to sponsored search.

3.2.1 Term Coupling Index (CTInd)

In the first indexing scheme, we index units that are composed of $\langle creative, term\rangle$ pairs $\langle c, t\rangle$. This is the most fine-grained indexing unit, and this approach effectively indexes the Cartesian product of the creatives and terms in each ad group. Since we index all possible $\langle creative, term\rangle$ pairs, the retrieval with this index is a single-level process — no postprocessing is required, since indexing units correspond to displayable ads. As we shall see below, other (more compact) indexes require some additional ranking after the initial retrieval is performed. The **CTInd** index, on the other hand, only requires filtering by gId_t, as we retrieve a single displayable ad per ad group (deduping). Algorithm 1 shows the retrieval algorithm pseudocode.

Index	Indexing unit	# indexing units	# indexed fields
CTInd	$\{\langle c,t\rangle : c \in \mathbf{C}^g, t \in \mathbf{T}^g, g \in \mathbf{G}\}$	$\lvert\mathbf{G}\rvert\overline{\lvert\mathbf{T}^g\rvert\lvert\mathbf{C}^g\rvert}$	$2\lvert\mathbf{G}\rvert\overline{\lvert\mathbf{T}^g\rvert\lvert\mathbf{C}^g\rvert}$
CrtvInd	$\{\langle c,\mathbf{T}^g\rangle : c \in \mathbf{C}^g, g \in \mathbf{G}\}$	$\lvert\mathbf{G}\rvert\overline{\lvert\mathbf{C}^g\rvert}$	$\lvert\mathbf{G}\rvert\overline{\lvert\mathbf{C}^g\rvert}(1+\overline{\lvert\mathbf{T}^g\rvert})$
AdGrpInd	$\{\langle \mathbf{C}^g,\mathbf{T}^g\rangle : g \in \mathbf{G}\}$	$\lvert\mathbf{G}\rvert$	$\lvert\mathbf{C}\rvert + \lvert\mathbf{T}\rvert$

Table 1: Summary of the three index versions.

Algorithm 1 CTRank (**CTInd** index)

1: $\langle \mathbf{c},\mathbf{t}\rangle \leftarrow [u_{(1)},\ldots,u_{(K)}] \subseteq \mathbf{C}\times\mathbf{T}$
2: FILTER $(\mathbf{c},\mathbf{t})$ BY gId_t
3: **return** $\langle \mathbf{c},\mathbf{t}\rangle$

In the **CTInd** index, the average number of indexing units per ad group is a product of cardinalities $\overline{|\mathbf{C}^g||\mathbf{T}^g|}$, and each such indexing unit contains two fields (i.e., a creative and term field). The expected number of fields indexed by the **CTInd** index is, therefore, $2|\mathbf{G}|\overline{|\mathbf{C}^g||\mathbf{T}^g|}$.

3.2.2 *Creative Coupling Index (***CrtvInd***)*

The indexing unit in this index is a single creative c coupled with all the bid terms associated with its ad group gId_c. This is a much smaller index that **CTInd**, since it no longer computes a Cartesian product of every creative and every term. Using this index, in order to retrieve a $\langle creative, term\rangle$ pair $\langle c,t\rangle$, we first retrieve a ranked list of creatives, filter them by gId_c and then retrieve the term with the highest score associated with each creative in the index; the score is computed according to Eq. 2. Algorithm 2 shows the retrieval algorithm pseudocode.

Algorithm 2 CrtvRank (**CrtvInd** index)

1: $\mathbf{c} \leftarrow [u_{(1)},\ldots,u_{(K)}] \subseteq \mathbf{C}$
2: FILTER $\mathbf{c}$ BY gId_c
3: $\mathbf{t} \leftarrow [\text{argmax}_{t\in gId_c}\, sc_q(t) : c \in \mathbf{c}]$
4: **return** $\langle \mathbf{c},\mathbf{t}\rangle$

In the **CrtvInd** index, for each creative we index, on average, $1+\overline{|\mathbf{T}^g|}$ term fields, and there are total of $|\mathbf{G}|\overline{|\mathbf{C}^g|}$ creatives in the collection. The expected number of indexed fields in this index is, therefore, $|\mathbf{G}|\overline{|\mathbf{C}^g|}(1+\overline{|\mathbf{T}^g|})$.

3.2.3 *Ad Group Coupling Index (***AdGrpInd***)*

In this approach, the indexing unit is the ad group itself. In order to retrieve a $\langle creative, term\rangle$ pair, first a ranked list of ad groups is retrieved. Then, for each ad group, a creative and a term with the highest scores are retrieved. Since the creative and the term scores are independent, and assuming that the scoring function is monotonic[3], the retrieved $\langle c,t\rangle$ pair is the pair with the highest score in the ad group. The algorithm pseudocode is presented in Algorithm 3.

Note that the **AdGrpInd** index is the only index type with no duplicated fields. The number of indexing units is the same as the number of ad groups, $|\mathbf{G}|$, and for each ad group there are, on average, $\overline{|\mathbf{T}^g|}+\overline{|\mathbf{C}^g|}$ fields. Therefore, the number of indexed fields is $|\mathbf{G}|(\overline{|\mathbf{T}^g|}+\overline{|\mathbf{C}^g|})$, which is also the number of unique fields in the ad corpus, $|\mathbf{C}|+|\mathbf{T}|$, as shown in Eq. 1.

This is the most compact index of the three alternatives we explore. Note also that this indexing scheme eliminates the need for filtering the results by the ad group identifier gId_u, since by definition each retrieved ad will be constructed from a different ad group.

[3]A scoring function is monotonic if each query term match has a positive contribution to the overall score.

Algorithm 3 AdGrpRank (**AdGrpInd** index)

1: $\mathbf{g} \leftarrow [u_{(1)},\ldots,u_{(K)}] \subseteq \mathbf{G}$
2: $\mathbf{c} \leftarrow [\text{argmax}_{c\in gId_g}\, sc_q(c) : g \in \mathbf{g}]$
3: $\mathbf{t} \leftarrow [\text{argmax}_{t\in gId_g}\, sc_q(t) : g \in \mathbf{g}]$
4: **return** $\langle \mathbf{g},\mathbf{c},\mathbf{t}\rangle$

4. STRUCTURED RERANKING

In the previous section, we discussed the basic ranking algorithms for each of the proposed indexing structures. In this section, we show that the basic ranking is sometimes not sufficient to produce the best retrieval results. As the size of each indexing unit grows, which is the case with the **CrtvInd** and **AdGrpInd** indexes, we face challenges that hinder the performance of the retrieval models. To address these challenges, which are described in Section 4.1, we propose a *structured reranking* strategy in Section 4.2.

4.1 Motivation for Structured Reranking

In the indexing strategies described in Section 3.2, the indexed and retrieved units are hierarchically structured from atomic and composite fields. Previous work on structured document retrieval shows that a combination of field scores often yields better retrieval performance than matching each field independently [7, 20, 2].

To test this hypothesis, we design a simple experiment that combines the scores obtained for the ad group and the $\langle creative, term\rangle$ pair retrieved by **AdGrpRank** (Algorithm 3). Formally, ads are *reranked* based on a mixture of scores:

$$sc_q(g,c,t) = \lambda sc_q(u_{c,t}) + (1-\lambda)sc_q(g), \qquad (3)$$

where $u_{c,t}$ is a $\langle creative, term\rangle$ pair $\langle c,t\rangle$ treated as a single indexing unit, and $sc_q(\cdot)$ is as defined in Eq. 2.

Fig. 2 shows the retrieval performance of this reranking (as measured by NDCG@10 [11]). Setting $\lambda = 0$ in Eq. 3 produces ranking that is equivalent to that of **AdGrpRank**, while setting $\lambda = 1$ produces ranking that ignores the ad group structure, and is, therefore, equivalent to that of **CTRank**.

It should be noted that in contrast to previous work, where mixture models usually yield improvements [7, 20, 2], combining the $\langle creative, term\rangle$ pair score with the ad group score is detrimental — setting $\lambda = 1$, and ignoring the ad group structure, yields the best performance. We postulate that this is a result of several key traits that differentiate sponsored search from other information retrieval tasks.

First of all, sponsored search is characterized by the large variance in ad group lengths. While some ad groups contain only a few bid terms, others can have up to 1,000 bid terms

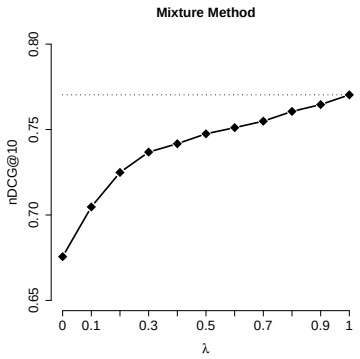


Figure 2: Mixture method retrieval performance with $0 \leq \lambda \leq 1$. Dotted line indicates the performance attained by CTRank with no mixture.

associated with them. This causes a strong length bias: the probability of shorter documents (ad groups with smaller number of terms) to be retrieved is higher than their probability of being relevant. While length bias is a well known phenomenon in information retrieval [27], its effects are more pronounced in sponsored search, since the latter has a much higher variance of document lengths.

Another issue that is unique for sponsored search is the cohesiveness of the ad groups. In traditional retrieval, it is usually assumed that documents, even structured ones, are about a single topic; however, this assumption does not necessarily hold for ad groups. A set of bid terms associated with an ad group can range from being focused and cohesive (associated with a single service or product) to being fragmented or even scattered (associated with several products or even with a broad set of generic bid terms), depending on the strategy of the advertiser. There is also a possibility that some advertisers might misuse the bidding mechanisms by purposefully associating unrelated terms to their ad groups in an attempt to attract more customers (spamming).

Since existing structured retrieval techniques do not address these issues, we propose a novel *structured reranking* approach that is specifically tailored towards sponsored search. Our approach relies on the ad structure and employs features that go beyond the simple mixture model in Eq. 3.

4.2 Structured Reranking Method

Our *structured reranking* method is a generalization of the mixture model presented in the previous section. It takes into account a given $\langle creative, term \rangle$ pair and the associated ad group. The method assumes we do an initial retrieval round using **AdGrpRank**. Then, the method reranks the initially retrieved ads using a linear model

$$rSc_q(g, c, t) \triangleq \sum_{i=1}^{n} \lambda_i f_i(g, c, t), \quad (4)$$

where $f_i(\cdot)$ is a feature function, λ_i are weights assigned to each function, and n is the number of such functions used for the reranking.

We refer to this reranking procedure as **StructRank**, and the entire retrieval procedure is described in Algorithm 4. Clearly, the choice of features used in **StructRank** plays a crucial role in the resulting algorithm performance. Limiting the choice of features to field scores alone yields the mixture model in Eq. 3. Since this model does not succeed in outperforming the baseline method that uses no structural information (**CTRank**), we expand the set of features used by **StructRank** to address the issues specific to sponsored search that were described in Section 4.1. As we show in Section 5, using this expanded set of features results in substantial performance improvements on a number of tasks.

Algorithm 4 StructRank (**AdGrpInd** index)

1: $\langle \mathbf{g}, \mathbf{c}, \mathbf{t} \rangle \leftarrow$ **AdGrpRank**
2: SORT DESC $\langle \mathbf{g}, \mathbf{c}, \mathbf{t} \rangle$ BY $rSc_q(g, c, t)$
3: **return** $\langle \mathbf{g}, \mathbf{c}, \mathbf{t} \rangle$

4.2.1 Reranking Features

We now define the features used by **StructRank**, as well as its parameter optimization. Recall that the features have the form $f(g, c, t)$, and are therefore defined over a $\langle c, t \rangle$ pair and an ad group associated with it. The features used are as follows.

- `crtvTermPairScore` is the score of the given $\langle creative, term \rangle$ pair. It is equivalent to the $sc_q(u_{c,t})$ component in the mixture model in Eq. 3.
- `adGrpScore` is the score of the entire ad group, from which the $\langle creative, term \rangle$ pair was selected. It is equivalent to the $sc_q(g)$ component in the mixture model in Eq. 3.
- `adGrpTermCount` is the number of term fields in a given ad group. As previously mentioned, a number of bid terms associated with an ad group can vary significantly. Since document length can affect the document's prior probability of retrieval [27], we explicitly add the number of bid terms as a feature in our model.
- `adGrpEntropy` is the entropy of an ad group. The entropy is computed over the individual words of the ad group as $-\sum_{w \in g} p_g(w) \log p_g(w)$, where the probability of word w_i is computed using a maximum likelihood estimate $p_g(w_i) = \frac{tf_{w_i,g}}{\sum_{w_j \in g} tf_{w_j,g}}$. Following previous work, where entropy was found related to document heterogeneity [2], we use entropy as an estimate of ad group cohesiveness—ad groups with smaller entropy will tend to be more cohesive.
- `adGrpQueryCover` produces a real number $r \in [0, 1]$, such that r is the ratio of query words "covered" by any field in the ad group. It is common, especially for longer queries, that not all query terms will appear in the selected creative and term pair. The `adGrpQueryCover` feature help differentiate between the ad groups that achieve high relevance scores due to a disproportional repetition of a single query word (bid term spamming) and those that achieve high relevance scores due to a more comprehensive coverage of query words.
- `adGrp[Field]Ratio` is the fraction of fields of type `[Field]` in an ad group that match at least one query word. Intuitively, we expect that ad groups that have high field match ratios w.r.t. query, will yield more relevant $\langle creative, term \rangle$ pairs, since these ad groups will tend to be more focused on the query topic. `[Field]` denotes either one of the sub-fields of the creative field, or the entire term field. This produces three features based on the location of the match: `adGrpURLRatio`, `adGrpTitleRatio` and `adGrpTermRatio`.

Therefore, we have a total of 8 features. Of course, adding other features is possible, but we limit ourselves to this small well-defined set of features for the purpose of this study.

4.2.2 Reranking Optimization

In this section, we present our method for optimizing the free parameters λ_i in the ranking function in Eq. 4. When the number of free parameters is small (as, for instance, is the case in Eq. 3), it is possible to optimize the parameters using an exhaustive search over the parameter space. However, when more features are introduced, such an exhaustive search quickly becomes infeasible.

To address this problem, we rely on a large and growing body of literature on the learning to rank methods for information retrieval (see Liu [18] for a survey). Learning to rank methods allow effective parameter optimization for ranking functions with respect to various retrieval metrics, even when a number of free parameters is high.

In this work, we employ a simple yet effective learning to rank approach that directly optimizes the retrieval metric of choice (e.g., NDCG@k). It is easy to see that our ranking function is *linear* w.r.t. λ_i. Therefore, we make use of the coordinate ascent algorithm proposed by Metzler and Croft [19]. This algorithm iteratively optimizes a multivariate objective function (in our case, $rSc_q(g, c, t)$) by performing a series of one-dimensional line searches. It repeatedly cycles through each parameter λ_i, holding all other parameters fixed while optimizing λ_i. This process is performed iteratively over all parameters until the gain in the target metric is below a certain threshold.

Although we use the coordinate ascent algorithm primarily for its simplicity and efficiency, any other learning to rank approach that estimates the parameters for linear models can be used. Other possible learning to rank algorithms include ranking SVMs [12], SVM^{MAP} [32] or RankNet [6].

Due to the linearity of our ranking function, query dependent features (e.g., query length) cannot be readily incorporated into **StructRank**, since they will have the same contribution across all documents associated with a query. While this can be addressed by using non-linear rankers, such rankers typically require more training data and are more prone to overfitting than linear models [31]. As a middle ground between linear and non-linear approaches, we bin our queries, and train a specific model for each bin. Any query-dependent feature (or combination of thereof) can be used for query binning. In our experiments we found that binning by query length is both conceptually simple and empirically effective for retrieval optimization.

5. EXPERIMENTS

In this section we describe the experimental results of our work. In Section 5.1 we describe the experimental setup. In Section 5.2 we empirically demonstrate the differences in index sizes and query run-times between the three ad index versions described in Section 3. In Section 5.3 we compare the retrieval effectiveness of these three index versions, as well as demonstrate the benefits of the structured reranking approach proposed in Section 4. Finally, in Section 5.4, we show the benefits of employing the ad structure for ad filtering.

5.1 Experimental Setup

All indexing and retrieval experiments are implemented using an open-source search engine Indri[4]. Indri natively supports indexing and retrieval of structured documents [28], and provides a best-in-class ad hoc retrieval performance using a popular language modeling approach for information retrieval [21]. Indri allows us to compare the performance of all our retrieval methods using the same search engine.

We constructed our dataset as follows. We sampled a set of queries from a Web search log using stratified sampling. We then randomly divided the queries into three disjoint sets: a development set of 1,570 queries used for debugging, feature selection and parameter tuning; a training set of 3,134 queries, and a held-out testing set of 773 queries with at least one relevant (non-*Bad*, see below) judgment associated with each query. We then retrieved ads for these queries using a commercial ad retrieval system, and had them evaluated by human judges using graded relevance judgments. This resulted in a set of judged query-ad pairs, and our experiments then consisted of *reranking* these pairs using each of the proposed methods. Each judged ad consisted of a single $\langle creative, term \rangle$ pair, and one such pair per ad group was judged. Overall, our collection consisted of 5,477 queries and 93,632 unique ad groups that contained a total of 239,011 creative fields and 5,143,010 term fields.

In all our retrieval experiments, all documents and queries were stemmed using Krovetz stemmer, and no stopword removal was performed. The smoothing parameter μ in Eq. 2 is set to 90, which was observed to be the optimal setting on the development set.

In the (re)ranking experiments, we measured the performance using the normalized discounted cumulative gain measure (NDCG), a standard retrieval metric for Web retrieval [11]. The relevance grades assigned by human judges were *[Perfect, Excellent, Good, Fair, Bad]*, and we used the following DCG gains for these grades *[10, 7, 3, 0.5, 0]*, respectively. In the parameter optimization of our reranking model, we use normalized DCG at position 10 as the target metric for the coordinate ascent search (Section 4.2.2).

As described in Section 4.2.2, we found that performance of the **StructRank** algorithm can be improved by separating the training set into several query bins, and training a separate linear model for each bin. Ideally, each bin should contain only queries of a certain type (for instance, only navigational or informational queries). In the lack of explicit query type information, we simply bin queries by length. We use three similar-sized bins, one for one word queries, one for two and three word queries and one for queries of four words or longer. We found that this binning procedure significantly improves performance on the development set, and used it in the rest of our ranking experiments.

5.2 Retrieval Efficiency

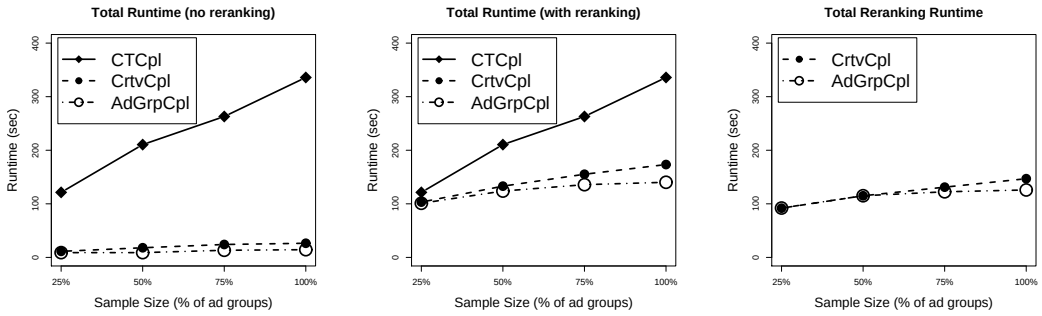
5.2.1 Index sizes

To test the efficiency of our proposed indexing strategies, we create 4 sub-samples of the increasing size from the entire corpus of 93,632 ad groups. Sub-sample sizes ranged between 25% and 100% of the entire corpus. Each of these subsamples was separately indexed using one of the three indexing methods: **CTInd**, **CrtvInd** and **AdGrpInd**. Overall, the combination of 4 sub-sample sizes and 3 index types yielded 12 experimental indexes. Table 2 details the number of documents in each sub-sample, and the size of all the indexes constructed. As can be seen from Table 2, index **AdGrpInd** is, as expected, the smallest of the three versions, both in terms of the number of documents, and the

[4] http://www.lemurproject.org/indri/

	Sample Size = 25%		Sample Size = 50%		Sample Size = 75%		Sample Size = 100%	
	# Docs	Index Size	# Docs	Index Size	# Docs	Index Size	# Docs	Index Size
CTInd	4,394,022	3.3G	8,662,762	6.4G	12,525,841	9.2G	16,814,306	13G
CrtvInd	59,636	223M	120,538	429M	178,025	616M	239,002	820M
AdGrpInd	23,367	91M	46,712	167M	69,656	240M	93,632	317M

Table 2: Details of index samples.



(a) (b) (c)

Figure 3: Retrieval runtime for the three index types. (a) No reranking; (b) with reranking of the top 10 results for CrtvInd and AdGrpInd using the `crtvTermPairScore` feature; (c) the difference between reranking and no-reranking runtimes.

overall index size. This is due to the fact that it is the only version of the index where no field duplications occur.

Due to a large number of terms that can be potentially associated with a single ad group (average number of terms per ad group is approximately 50, with maximum number of terms per ad group bounded by 1,000), the **CTInd** index is the most costly, both in terms of number of documents and the index size. On the smallest 25% sub-sample, the size of index **CTInd** is 36 times larger than that of the index **AdGrpInd**. Moreover, this difference grows with the number of ad groups, as the ad group fields in index **CTInd** are not uniquely indexed. On the entire corpus, the size of index **CTInd** is already 41 times larger than that of index **AdGrpInd**. Note that in our experiments, only a small number of ad groups ($\sim 100,000$) is used. As the ad corpora used by commercial search engines typically contain millions of ad groups and are frequently updated, constructing the $\langle creative, term \rangle$ pairs index **CTInd** becomes practically infeasible.

The same trend, though on a smaller scale, occurs when comparing the **CrtvInd** index and the **AdGrpInd** index. Each ad group contains, on average, 2.5 creatives, and the **CrtvInd** strategy indexes each of the terms once for each of them. As can be seen in Table 2, the same ratio of 2.5 approximately explains the difference in index size between **CrtvInd** and **AdGrpInd**.

5.2.2 Query Runtime

Based on the relative sizes of the indexes, we expected to see a significant reduction in query runtime when **AdGrpInd** is used, especially compared to **CTInd**. To test this, we sample 900 queries from our development set, and run them on all 12 indexes, with or without reranking, as shown in Fig. 3. Each curve corresponds to an index type; each point on the curve corresponds to one of the sub-samples.

It is evident that without the reranking step the execution of the queries in **CTInd** is significantly slower than that of the other two methods (Fig. 3 (a)). Moreover, the difference in runtime increases with the index size. Reranking the top ten results (Fig. 3 (b)) increases the runtime for the two indexes where reranking is performed[5], however the runtime still remains well below of that of **CTInd**. Finally, Fig. 3 (c), shows the difference between the runtime with or without reranking for **CrtvInd** and **AdGrpInd**. Note, that this difference increases slowly with the size of the index. This growth is much slower than that demonstrated by **CTInd**, indicating that even for larger index sizes, reranking with the **CrtvInd** or **AdGrpInd** indexes will remain a more efficient strategy than retrieval with the **CTInd** index.

5.3 Ad Ranking

In the previous section, we have shown that the **AdGrpInd** index provides the smallest index size and the best retrieval efficiency, overall. In this section, we demonstrate that when the proposed structured reranking algorithm **StructRank** is applied, **AdGrpInd** can provide the best retrieval effectiveness (in terms of nDCG), as well.

To this end, we consider the *ad ranking* task. This task is similar to the standard ad hoc information retrieval setting. Given a user query, ads are ranked by their relevance score and the top ones are presented to the user. Note, that in contrast to standard document retrieval, only the selected $\langle creative, term \rangle$ pair is shown to the user, and not the entire ad group.

Our retrieval algorithms differ in the way the underlying index is structured and in the way the relevance score is computed (see Section 3). We evaluate all the algorithms using a set of judged query-document pairs originally retrieved by a commercial ad retrieval system[6]. All the methods are evaluated using a held-out set of 773 queries that have at least one

[5]Recall that reranking is not needed for the **CTInd** index.

[6]We refrain from comparing our experimental system to the production system, since it takes into account many other features not accounted for in this study.

	nDCG@1	nDCG@5	nDCG@10
AdGrpRank	0.570	0.631	0.668
CrtvRank	0.577	0.654	0.698
CTRank	0.655^{*}	0.725^{*}	0.767^{*}
StructRank	$\mathbf{0.679}^{*}_{\dagger}$ *(+3.66%)*	$\mathbf{0.742}^{*}_{\dagger}$ *(+2.34%)*	$\mathbf{0.780}^{*}_{\dagger}$ *(+1.69%)*

Table 3: Retrieval effectiveness of all the retrieval methods. The '$*$' symbol denotes statistically significant difference between the results of CTRank and those of AdGrpRank and CrtvRank methods, while the '†' symbol denotes statistically significant difference between StructRank and CTRank (using Wilcoxon sign test, $\alpha < 0.05$). The numbers in the parentheses indicate % improvement of StructRank over the CTRank baseline.

non-*Bad* judgment associated with them (that is, the ranking in response to this queries can be potentially improved by *reranking*). We employ each of the retrieval strategies described in Section 3 as baselines. In addition, we use the **StructRank** algorithm (Algorithm 4) to test whether structural information captured by this method benefits the ranking.

Table 3 summarizes the retrieval results for all four retrieval algorithms. First, we note that **CTRank** has the best performance among the baseline methods. **CTRank** outperforms the other baselines to a statistically significant degree on all nDCG measures. While using **CTRank** for *ranking* using a **CTInd** index is infeasible for large corpora (as was explained in Section 5.2), Table 3 shows that it can significantly improve the performance when applied as a *reranking* method on a smaller index.

The key result in Table 3 is the performance of the **StructRank** algorithm. As described in Section 4.2, **StructRank** ranking function encodes the hierarchical structure of an ad group as a vector of features, and linearly combines these features. **StructRank** consistently and significantly improves the performance over the **CTRank** baseline at all NDCG positions. The improvements at the top ranks are particularly high (3.7% improvement for nDCG@1); this is crucial for sponsored search since the user only sees a few top results (ads).

Fig. 4 plots the nDCG curves corresponding to the four retrieval algorithms from Table 3. The performance improvements attained by **StructRank** over the baselines are consistent over ranks 1 through 10.

As mentioned in Section 5.1, **StructRank** learns a separate linear model for each query length bin. Table 4 provides a breakdown of retrieval performance improvements attained for each of these bins. Table 4 shows consistent improvements over the best baseline, **CTRank**, in all the bins. These performance gains are the most striking for long (four words or more) queries, reaching more than 7% improvement for NDCG@1.

This indicates the importance of using the structural information of ad groups for longer queries, as the probability of an exact or even partial match of these queries to any single $\langle creative, term \rangle$ pair in the index is low. Therefore, the ad group structure provides a much needed context to improve the coverage and the relevance of the ads displayed in response to these long queries.

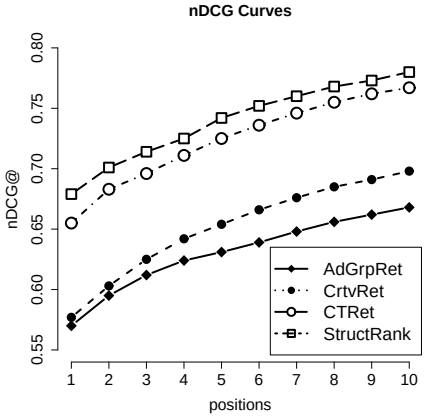


Figure 4: NDCG Curves.

Finally, Table 5 shows the highest weighted features, as learned for each of the query length bins. Note the dominantly high weight of a `crtvTermPairScore` feature, and its increase with the query length. This indicates the high importance of a close match with a $\langle creative, term \rangle$ pair, if such exists, especially for long queries. This high weight also explains the success of the **CTRank** algorithm, which significantly outperforms the other two baseline methods (see Table 3).

In addition, we note the differences in feature weights learned for each bin. While for short, single-word queries, features traditionally associated with navigational queries (like title and URL match) receive high weights, for longer queries, query word coverage features (for instance, the feature `adGrpTermQueryCover`) are more important. These differences showcase both the benefit of our strategy of learning a separate model for each query bin, and an adequacy of query length as the binning criterion.

5.4 Ad Filtration

In this section we apply our algorithms to the *ad filtration* task [24]. In this task, the goal is to learn a binary classifier that is trained to distinguish between *relevant* and *non-relevant* ads. This type of classifier is often applied to an initial ranked list of ads that has been returned by an information retrieval system such as the one described in the previous section.

The classifier is used to pass only the most relevant ads to the next stage of ranking, where they can be reranked using non-textual features such as click-through rates and bid amounts. Filtration of non-relevant ads is important in sponsored search, where the display of an inappropriate ad can lead to a decrease of click-through rate on the entire list. On the other hand, a failure to show a highly relevant ad can lead to a loss of potential revenue for the search engine.

Accordingly, we convert the original graded relevance judgments into a binary class label. All the $\langle creative, term \rangle$ pairs rated as *Bad* w.r.t. a query are assigned a *negative* label, while all the other pairs are assigned a *positive* label. As is often the case with relevance judgments in information retrieval, the majority of our data is associated with the negative class. Out of 27,763 rated pairs in our development set, only 6,219 pairs have a positive label.

	Len1 (143 queries)			*Len2+3* (443 queries)			*Len4+* (187 queries)		
	nDCG@1	nDCG@5	nDCG@10	nDCG@1	nDCG@5	nDCG@10	nDCG@1	nDCG@5	nDCG@10
CTRank	0.765	0.841	0.865	0.655	0.716	0.757	0.569	0.656	0.715
StructRank	**0.803** *(+4.96%)*	**0.849** *(+0.95%)*	**0.871** *(+0.69%)*	**0.669** *(+2.14%)*	**0.731*** *(+2.09%)*	**0.769*** *(+1.59%)*	**0.610** *(+7.2%)*	**0.686*** *(+4.57%)*	**0.737*** *(+3.08%)*

Table 4: Retrieval effectiveness of retrieval methods CTRank and StructRank, for queries of different length. The $*$ denotes statistically significant difference with CTRank method (Wilcoxon sign test, $\alpha < 0.05$). The numbers in the parentheses indicate % improvement of StructRank over CTRank baseline.

Len1		*Len2+3*		*Len4+*	
`crtvTermPairScore`	0.83	`crtvTermPairScore`	0.93	`crtvTermPairScore`	0.95
`adGrpUrlMatchRatio`	0.05	`adGrpEntropy`	0.03	`adGrpTermQueryCover`	0.03
`adGrpEntropy`	0.04	`adGrpScore`	0.02	`adGrpTermCount`	< 0.01
`adGrpTitleMatchRatio`	0.03	`adGrpTermQueryCover`	0.02	`adGrpEntropy`	< 0.01
`adGrpTermQueryCover`	0.02	`adGrpUrlMatchRatio`	< 0.01	`adGrpUrlMatchRatio`	< 0.01

Table 5: Learned weights for different query groups.

	Precision	Recall	F1
AdGrpRank	0.510	0.399	0.448
CrtvRank	0.596	0.458	0.518
CTRank	0.612	0.560	0.585
StructRank	**0.680** *(+11.1%)*	**0.660** *(+17.9%)*	**0.670** *(+14.5%)*

Table 6: Summary of ad filtration results. Precision and recall are reported at the decision threshold that achieves maximum F1.

We use C4.5 decision tree implemented in Weka for all our classification experiments [30]. In the development phase of our experiments, we tested other linear and non-linear classifiers implemented in Weka, and found that C4.5 decision tree performed better than other stand-alone classifiers, and its performance was comparable to that obtained by complex ensemble classifiers like AdaBoost, while being much faster at train time.

The class imbalance present in our data often leads to a poor recall of positive examples. Since Weka allows to output a prediction confidence level for each instance, we tune the prediction threshold of our classifier (a confidence level below which the instance is classified as negative) to optimize the F1 measure on the development set.

Previous work [24] concentrated on identifying the features *within* the $\langle creative, term\rangle$ pair that improve the ad filtration performance. In this work, we are interested in the importance of features *outside* the $\langle creative, term\rangle$ pair, namely the structural features from the entire ad group, as discussed in Section 4.2.1. To this end, we construct three baseline classifiers, each using as a single feature an output of the retrieval algorithms **CTRank**, **CrtvRank** and **AdGrpRank** ($\langle creative, term\rangle$ pair, creative and ad group scores, respectively). We compare the performance of these baseline classifiers, to the performance of a classifier which uses all the structural features used by **StructRank**.

Table 6 shows the precision, recall and F1 measure attained by the three baseline classifiers and the structural classifier, denoted **StructRank**, for an ad filtration task. As Table 6 demonstrates, **StructRank** is superior to all the baselines. It achieves more than 10% improvement over the best performing baseline, **CTRank**, on all three measures, and all the improvements are statistically significant according to the McNemar's test ($\alpha < 0.05$).

Table 7 provides a more detailed analysis of the ad filtration experiments. Table 7 (a) breaks down the performance of each classifier (in terms of F1 measure) by query length bins, similarly to the analysis shown in Table 4 for the NDCG measure. The key observation in this table is the importance of structural features for long queries. While the contribution of these features is significant for all query lengths, for long queries (having with four or more words) they provide an almost 90% improvement (sic!) over the baselines that use no structural information.

Finally, Table 7 (b) details the filtration rates for all five relevance grades. It is clear that **StructRank** consistently outperforms the other baselines, filtering out *fewer* relevant and *more* non-relevant ads, across the entire range of relevance grades.

6. CONCLUSIONS

In this paper we investigated the important yet often overlooked issue of ad indexing. Existing advanced match algorithms index the ads using standard IR indexing techniques with flat document representation. However, in practice ads are inherently hierarchically structured and organized into accounts, campaigns and ad groups. We investigated three different strategies for indexing this structured data corpus, including creative-bid term coupling, creative coupling, and ad group coupling. We showed that the creative-bid term coupling, which is the most straightforward implementation, results in an infeasibly large index, while the ad group coupling index was both significantly smaller and much more practically useful.

In addition to investigating different indexing strategies, we also explored how to effectively retrieve ads using structured ad indexes. We proposed a novel reranking strategy that exploits the ad corpus structure. Our structured reranking approach makes use of a linear machine-learned ranking function that uses a variety of structural ad features. As we show, the approach is not only useful for ranking ads, but also for filtering (classifying) bad ads.

We conducted a comprehensive set of experiments on a sponsored search test collection from a large commercial search engine. Our experimental results show that using the

	Len1 (2419)	*Len2+3* (16182)	*Len4+* (9162)
AdGrpRank	0.636	0.427	0.286
CrtvRank	0.667	0.518	0.343
CTRank	0.750	0.571	0.272
StructRank	**0.834** *(+11.2%)*	**0.664** *(+16.3%)*	**0.516** *(+89.7%)*

(a)

Grade	**AdGrpRank**	**CrtvRank**	**CTRank**	**StructRank**
Perfect (54)	0.574	0.204	0.278	0.204
Excellent (71)	0.437	0.380	0.268	0.225
Good (980)	0.542	0.419	0.339	0.199
Fair (5114)	0.614	0.572	0.463	0.370
Overall Rel. (6219)	*0.601*	*0.542*	*0.440*	*0.340*
Bad (21544)	0.889	0.910	0.897	0.910

(b)

Table 7: (a) F1 measures for the ad filtration classifiers, binned by query length. (b) Detailed filtration rates (ratio of filtered ads) for each relevance grade. The numbers in parentheses indicate the number of $\langle creative, term \rangle$ pairs in each group.

proposed structured reranking strategy with the (small) ad group index yields statistically significant improvements in retrieval and filtering effectiveness over the simple approach that produces a combinatorially huge creative-bid term index. To summarize, we show that a small, structured index can be used to retrieve highly relevant sponsored search ads.

All of the experiments reported in this paper were conducted on a sponsored search test collection. However, we believe that our methodology can be directly applied to the content match scenario as well. Although the construction of the ad query would be different in that case (the input would be a Web page rather than a search query), the server-side indexing and retrieval of ads should mostly stay the same as in sponsored search. We intend to apply our methodology to content match advertising in future work.

7. REFERENCES

[1] A. Anagnostopoulos, A. Z. Broder, E. Gabrilovich, V. Josifovski, and L. Riedel. Just-in-time contextual advertising. In *Proceedings of CIKM*, pages 331–340, 2007.

[2] M. Bendersky and O. Kurland. Utilizing Passage-Based Language Models for Document Retrieval. In *Proceedings of ECIR*, pages 162–174, 2008.

[3] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *Proceedings of WWW*, pages 511–520, 2009.

[4] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proceedings of SIGIR*, pages 559–566, 2007.

[5] A. Z. Broder, P. Ciccolo, M. Fontoura, E. Gabrilovich, V. Josifovski, and L. Riedel. Search advertising using web relevance feedback. In *Proceedings of CIKM*, 2008.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of ICML*, 2005.

[7] J. P. Callan. Passage-level evidence in document retrieval. In *Proceedings of SIGIR*, pages 302–310, 1994.

[8] W. B. Croft and J. Lafferty. *Language Modeling for Information Retrieval (The Information Retrieval Series)*. Springer, December 2003.

[9] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: a system for personal information retrieval and re-use. In *Proceedings of SIGIR*, pages 72–79, 2003.

[10] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlavik. *Advances in XML Information Retrieval*. Springer, 2005.

[11] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM TOIS*, 20(4):422–446, 2002.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, pages 133–142, 2002.

[13] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. of WWW*, pages 387–396, 2006.

[14] G. Kazai and A. Doucet. Overview of the INEX 2007 book search track: Booksearch '07. *SIGIR Forum*, 42(1):2–15, 2008.

[15] A. C. König, K. Church, and M. Markov. A data structure for sponsored search. In *Proceedings of ICDE*, 2009.

[16] A. Lacerda, M. Cristo, M. A. Gonçalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *Proceedings of SIGIR*, pages 549–556, 2006.

[17] M. Lalmas. XML information retrieval. In *Encyclopedia of Library and Information Sciences*. Taylor and Francis Group, 2009.

[18] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3), 2009.

[19] D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[20] P. Ogilvie and J. Callan. Hierarchical language models for XML component retrieval. In *Advances in XML Information Retrieval*, pages 224–237. Springer, 2005.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281, 1998.

[22] X. Qi and B. D. Davison. Classifiers without borders: incorporating fielded text from neighboring web pages. In *Proceedings of SIGIR*, pages 643–650, 2008.

[23] F. Radlinski, A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, and L. Riedel. Optimizing relevance and revenue in ad search: A query substitution approach. In *Proceedings of SIGIR*, 2008.

[24] H. Raghavan and D. Hillard. A relevance model based filter for improving ad quality. In *Proceedings of SIGIR*, 2009.

[25] B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. S. de Moura. Impedance coupling in content-targeted advertising. In *Proceedings of SIGIR*, 2005.

[26] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of CIKM*, pages 42–49, 2004.

[27] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing and Management*, 32(5):619–633, 1996.

[28] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*, 2004.

[29] H. Wang, Y. Liang, L. Fu, G.-R. Xue, and Y. Yu. Efficient query expansion for advertisement search. In *Proceedings of SIGIR*, pages 51–58, 2009.

[30] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.

[31] Q. Wu, C. Burges, K. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, pages 1–17, 2009.

[32] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of SIGIR*, pages 271 – 278, 2007.

[33] C. Zhai. Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.

[34] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Inf. Sys. (TOIS)*, 22(2):179–214, 2004.