Efficient Query Expansion for Advertisement Search

Haofen Wang, Yan Liang, Linyun Fu, Gui-Rong Xue, and Yong Yu
Dept. of Computer Science & Engineering, Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai, China
{whfcarter, yliang, fulinyun, grxue, yyu}@apex.sjtu.edu.cn

ABSTRACT

Online advertising represents a growing part of the revenues of major Internet service providers such as Google and Yahoo. A commonly used strategy is to place advertisements (*ads*) on the search result pages according to the users' submitted queries. Relevant ads are likely to be clicked by a user and to increase the revenues of both advertisers and publishers. However, bid phrases defined by ad-owners are usually contained in limited number of ads. Directly matching user queries with bid phrases often results in finding few appropriate ads. To address this shortcoming, *query expansion* is often used to increase the chances to match the ads. Nevertheless, query expansion on top of the traditional inverted index faces efficiency issues such as high time complexity and heavy I/O costs. Moreover, precision cannot always be improved, sometimes even hurt due to the involvement of additional noise.

In this paper, we propose an efficient ad search solution relying on a block-based index able to tackle the issues associated with query expansion. Our index structure places clusters of similar bid phrases in corresponding blocks with their associated ads. It reduces the number of merge operations significantly during query expansion and allows sequential scans rather than random accesses, saving I/O costs. We adopt flexible block sizes according to the clustering results of bid phrases to further optimize the index structure for efficient ad search. The pre-computation of such clusters is achieved through an agglomerative iterative clustering algorithm. Finally, we adapt the spreading activation mechanism to return the top-k relevant ads, improving search precision. The experimental results of our prototype, AdSearch, show that we can indeed return a larger number of relevant ads without sacrificing execution speed.

Categories and Subject Descriptors: H.3.1 [Content Analysis and Indexing]: Indexing methods; H.3.3 [Information Search and Retrieval]: Search process, Clustering

General Terms: Algorithms, Design, Experimentation, Performance

Keywords: Ad Search, Efficient Query Expansion, Block-based Index, Agglomerative Clustering, Ad Rank

SIGIR'09, July 19–23, 2009, Boston,Massachusetts,USA.

1. INTRODUCTION

Over the years, the Web has become an important venue for advertising, where search engine companies, such as Google, Yahoo and Baidu, are among the biggest advertisement (*ad*) publishers. There are mainly two kinds of advertising channels: contextual advertising and sponsored search. The former places ads on Web pages. Relevant keywords or patterns are extracted from the page content for ad selection and display during user browsing. As for the latter, a search engine often finds one or more ads that a user may be interested in after a query is issued. It displays them on the result page. The clicks on these ads may bring user traffic to the advertisers' Web sites and increase the publishers' revenues. The ranking of these ads is derived from both their relevance to the user's query or page content and auctions from the advertisers. As we do not have access to information from advertisers, we focus on relevance matching for sponsored search and do not consider placement strategies or auctions.

Ads are characterized by *bid phrases* that represent the keywords the advertisers choose for their ads. These bid phrases are used to match the user queries. Empirical evidence [21] has indicated that relevant ads to the user queries are more likely to be clicked. According to the statistical information presented in Section 7, the number of related ads with respect to the bid phrases queried by users follows the power law distribution. Most bid phrases are associated with few ads. As a result, syntactic approaches trying to directly match the input queries to the bid phrases often suffer from low recall when searching for ads. This is due to the fact that these methods simply return those ads containing certain bid phrases and fail to find potentially relevant ads. For example, when a user submits "job training" to a search engine based on this approach, a relevant ad containing "career college" does not syntactically match the query and is not proposed. Compared with document retrieval, the situation is even worse due to the shorter lengths of the ads and the sparsity of the bid phrase distribution over ads.

A. Z. Broder et al. [7] proposed to move down the "long tail" of a power law distribution curve through an aggregation of rare queries. Zhang et al. [22] also argued that the vocabulary used by bid phrases is often limited compared with that of user queries. To solve the problem, the mechanism of *query expansion* may help improve the recall as it will include other related bid phrases and thus provide a more complete answer set. A straightforward method for effective ad search is to perform query expansion on top of a well-optimized inverted index, where bid phrases are considered as index terms and ads treated as index documents. During query processing, the search engine locates several inverted lists for both original and expanded bid phrases, and then returns the relevant ads merged from these lists. However, efficiency is a problem as the large number of merge operations leads to the high time complex-

ity and numerous random accesses to these inverted lists result in heavy I/O costs. Moreover, precision cannot always be improved, sometimes even hurt due to the involvement of additional noise.

In this paper, we propose an efficient ad search solution which is able to tackle the issues associated with query expansion. Our main contributions are listed as follows.

- We present a block-based index where similar bid phrases are placed in a same block or continuous blocks with their associated ads. It reduces the number of merge operations significantly during query expansion. Besides, the ability to use a sequential access on these blocks further reduces the I/O overhead. We optimize the corresponding index structure for efficient ad search thanks to the use of flexible block sizes determined by the clustering results of bid phrases.
- We leverage an agglomerative iterative clustering algorithm to group bid phrases into clusters. Similar bid phrases in a same cluster are used to expand the original one to find a larger number of relevant ads.
- We adapt a spreading activation algorithm [13] to return the top-k relevant ads, improving the search precision. It propagates the relevance scores from similar bid phrases associated with given ads by considering the latent similarity between these bid phrases and the input keyword query.

We implemented our solution in a system called AdSearch and experimentally show that it achieves good performance on both Chinese and English datasets. These results indicate that the system can be used by search engine companies since it allows them to return a larger number of relevant ads in their online advertising programs without sacrificing execution speed.

The remaining of the paper is organized as follows. Section 2 introduces the related work. Section 3 presents the overview of AdSearch. Section 4 explains how to discover groups of similar bid phrases, Section 5 explains the block-based index structure and its construction method, and Section 6 shows how to find the top-k relevant ads. In Section 7, we present our experiments to demonstrate both efficiency and search quality. Finally, the paper is concluded in Section 8.

2. RELATED WORK

2.1 Overview of Advertisement Search

Web advertising has become an active research area in recent years. A. Broder [6] pointed out that finding the "best match" between a given user in a given context and a suitable advertisement involves a variety of optimization and search problems. There are two kinds of online advertising channels including sponsored search and contextual advertising. As for sponsored search, an overview is given in [15]. A business model for search engines in sponsored search has been discussed by B. Jansen in [17]. W. Zhang et al. [22] pointed out that Web searchers can issue any queries while bid phrases are limited. They focused on the query rewriting technique through active learning without tackling the efficiency problem. A. Broder et al. [7] confirmed that search advertising can benefit from aggregating queries in the "long tail" and they focused on the classification of these rare queries. M. Ciaramita et al. [12] investigated the sponsored search problem from a machine learning perspective, while F. Radlinski et al. [19] discussed a two-stage approach to optimize ad relevance calculation with low computational cost while at the same time improving the expected revenue. They used external knowledge in off-line preprocessing phase and developed an ahead-of-time query-to-query substitution system. The above listed work shed light on the importance of designing an efficient query expansion mechanism relying on a block-based index for ad search.

Recent work [8, 18, 20, 1, 11] dealt with the contextual advertising problem. A. Lacerda et al. [18] described a system that learned how to extract keywords from Web pages for advertisement targeting using a number of features. B. Ribeiro-Neto et al. [20] proposed several strategies to match Web pages with ads based on keyword extraction. A. Lacerda et al. [18] proposed a new framework to associate ads with web pages based on Genetic Programming (GP). A. Broder et al. [8] proposed a system for contextual advertising based on a combination of semantic and syntactic features. A. Anagnostopoulos et. al [1] discussed matching ads with dynamically created pages on the basis of text summarization techniques paired with external knowledge. D. Chakrabarti et al. [11] improved the matching between individual ads and the content of Web pages by considering the ad-page scoring function with extra parameters from a logistic regression model. The work of this kind focused on keyword extraction and page content understanding, which can be regarded as an additional step for our future work.

2.2 Index Structure

H. Bast and I. Weber [4] proposed a new index structure that uses no more space than a state-of-the-art compressed inverted index, but with much faster query processing speed for autocompletion search. Moreover, they presented in [3] an efficient realization of an interactive search engine. They built flat clusters of related terms and added this information as artificial words to the index such that good hits related to the original query could be suggested. Different from this work, we are the first to apply block-based index for efficient ad search. We place bid phrases together in a same or continuous blocks based on similarity rather than lexical closeness. These similar bid phrases are pre-computed into clusters by an agglomerative iterative algorithm. Additionally, we consider a more effective ranking scheme to return the most relevant ads through an adaptation of the spreading activation algorithm.

2.3 Query Expansion

Clustering methods are very useful to group related terms together. The traditional content-based clustering algorithms have been studied for years. I. S. Dhillon [14] presented the novel idea of modeling the document collection as a bipartite graph between documents and words and posed a bipartite partitioning problem for the simultaneous clustering problem. A similar method was used by D. Beeferman et al. [5] to discover clusters of similar user queries and similar URLs. H. Cao et al. [9] proposed a context-aware query suggestion approach based on an offline learning step which clusters a click-through bipartite graph. However, they did not focus on the clustering of ads and bid phrases and did not consider the characteristics of advertising data. J. J. Carrasco et al. [10] presented both top-down and bottom-up hierarchical clustering algorithms for the large bipartite graph of bid phrases and ads. The novelty of our method is that we consider the benefit of integrating clustering with indexing (i.e. the varying block sizes for further optimized ad search is determined by bid phrase clustering results).

3. ADSEARCH OVERVIEW

As shown in Figure 1, AdSearch consists of two main components. At the offline side, *Graph Constructor* first constructs a bipartite graph for bid phrases and ads, *Clustering Calculator* then computes clusters of bid phrases through an agglomerative clustering algorithm on this large graph, and *Index Builder* constructs the index by placing similar bid phrases (from the same cluster)

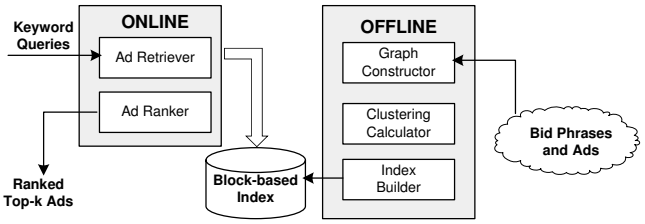


Figure 1: AdSearch Architecture

and their associated ads together in continuous blocks. At the online side, *Ad Retriever* expands a keyword query with related bid phrases and retrieves the associated ads through a single-pass scan on corresponding blocks of the index for efficient query expansion and finally *Ad Ranker* returns the top-k relevant ads based on an effective ranking scheme adapted from a spreading activation mechanism.

4. BID PHRASE CLUSTERING

4.1 Bipartite Graph Construction for Bid Phrases and Ads

We cluster bid phrases without analyzing the content of their associated ads. This is due to the fact that ads are rather different from documents in terms of lengths and intention, and the existing linguistic analyzers are language dependent. We make use of the structural characteristic between bid phrases and ads to accomplish this task. They build up a bipartite graph where the two independent sets are composed of unique bid phrases and unique ads, respectively. A bid phrase and an ad are connected if the bid phrase is contained in or associated with the ad. The bipartite graph construction is described by Algorithm 1.

Algorithm 1: The bipartite graph construction process

Input: a corpus of data C in the form of (bid phrase, ad) pairs.
Output: a bipartite graph G.

1 Collect a set B of unique bid phrases from C;
2 Collect a set A of unique ads from C;
3 For each unique bid phrase from B, create a vertex v_b in G;
4 For each unique ad from A, create a vertex v_a in G;
5 If bid phrase b is associated with ad a, add an undirected edge in G between the corresponding vertices v_b and v_a.

4.2 Agglomerative Iterative Clustering

Two bid phrases (or ads) are similar if they are associated with a large number of common ads (or bid phrases). The more their associated ads (or bid phrases) overlap, the more similar they are. More formally, we use Jaccard Similarity [16] and define function $\sigma(x, y)$ between two vertices x and y in the bipartite graph by

$$\sigma(x,y) = \begin{cases} \frac{|N(x) \bigcap N(y)|}{|N(x) \bigcup N(y)|} & \text{if } |N(x) \bigcup N(y)| > 0 \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

where $N(x)$ denotes the set of the neighbors of a vertex x. The similarity score lies in the range of $[0, 1]$, being 0 if the vertices have no neighbor in common, and 1 if they have exactly the same neighbors. The score between two vertices from different independent sets is defined to be zero. This similarity measure is efficient as it only considers the structure features of bid phrases. Figure 2 depicts the similarity calculation for the bid phrase pairs in an example bipartite graph. The similarity score between bid phrases A and B equals 0.25 as A is associated with the ads Ad0 and Ad3, and B connects to the ads Ad1, Ad2 and Ad3.

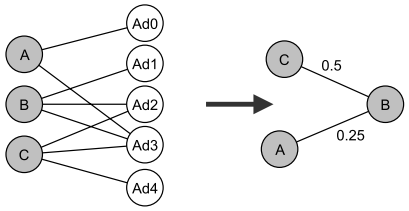


Figure 2: Similarity calculation according to the structure of an example bipartite graph.

One could simply cluster the vertices v_b in G to discover unions of similar bid phrases using $1 - \sigma$ as the distance metric. However, as pointed out by [5], this simplistic method suffers from a lack of semantic level similarity. Thus, we apply a slightly adapted agglomerative clustering algorithm to find groups of semantically related bid phrases. Bid phrases and ads are alternately clustered in the same way until the termination condition is met. It helps uncover latent relationships between bid phrases which are associated with very similar (but not exactly the same) ads. Algorithm 2 presents the detailed process.

Algorithm 2: The clustering process

Input: a bipartite graph G.
Output: a clustered bipartite graph G' where each vertex representing bid phrases in G' corresponds to one or more vertices of the same type in G.

1 Score all pairs of bid phrase vertices in G according to (1);
2 Merge the two vertices v_b, v'_b for which $\sigma(v_b, v'_b)$ is largest;
3 Score all pairs of ad vertices in G according to (1);
4 Merge the two vertices v_a, v'_a for which $\sigma(v_a, v'_a)$ is largest;
5 Go to Step 1 until the termination condition is met.

One reasonable termination condition can be defined as the threshold of the maximum similarity. A high threshold creates smaller clusters and thus fewer related bid phrases can be used for query expansion, while a low threshold places less relevant bid phrases into a single cluster, thus increasing the iteration count. The suitable threshold discussed in [10] is adopted and we continue clustering until

$$\max_{b_i, b_j \in B} \sigma(b_i, b_j) \leq 0.05 \text{ and } \max_{a_i, a_j \in A} \sigma(a_i, a_j) \leq 0.05. \quad (2)$$

After calculating $\sigma(x, y)$ once for all relevant vertex pairs, only a few $\sigma(x, y)$ values change at each iteration: after merging two vertices v_a and v'_a, the distance between v_b and v'_b changes only if

$$v_b \in N(v_a) \cup N(v'_a) \text{ and } v'_b \in N(v_a) \cup N(v'_a). \quad (3)$$

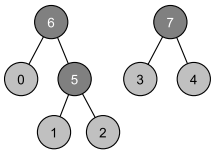
Let $|N|_{max}$ be the maximum number of neighbors of any vertex in G. It requires at most $2|N|_{max}$ distance calculations after each merge, which keeps the efficiency of the clustering procedure.

The resulting graph consists of several clusters of bid phrases. These similar phrases are put together in a same block or continuous blocks with the corresponding ads in the block-based index. The index structure will be described in Section 5 in detail.

5. INDEX STRUCTURE FOR EFFICIENT AD SEARCH

5.1 Mapping Clusters of Bid Phrases to Index Terms

The pre-computed clusters of bid phrases form a spanning forest in which each cluster is a binary tree and phrases or clusters merged in the same iteration are put at the same level. Each vertex in these trees has a unique ID. The leaves correspond to bid phrases while the internal vertices represent clusters. Figure 3 (a) shows a fragment of cluster trees in the forest. In this example, five bid phrases corresponding to vertices 0 to 4 belong to two cluster trees.



Vertex ID	Bid Phrase	Artificial Word
0	A	6:0
1	B	6_5:1
2	C	6_5:2
3	D	7:3
4	E	7:4

Figure 3: (a) Fragment of a cluster forest (b) Mapping table

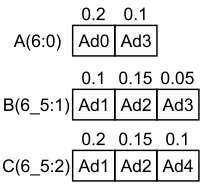
For each leaf in the forest, there exists a unique path from the root of the corresponding cluster tree to the bid phrase. More formally, assuming that a bid phrase c_0 belongs to the clusters $c_1, c_2, \cdots, c_p$ and $c_i \subset c_{i+1}$ for any $1 \leq i \leq p-1$, there exists the only path $c_p c_{p-1} \cdots c_1 c_0$ from the root c_p of the cluster tree to the bid phrase. Thus, each bid phrase in the cluster can be mapped into an artificial word $c_p\_c_{p-1}\_\cdots\_c_2\_c_1{:}c_0$ containing the information of the cluster path as its prefix. Figure 3 (b) illustrates the mapping results for our example bid phrases. For instance, bid phrase A is mapped to the artificial word "6:0". These artificial words record the real content of the bid phrases together with their cluster paths.

We treat artificial words as index terms and place them in the index according to the lexical order, which guarantees that words sharing the same prefix are adjacent to each other. This way, the mappings ensure that similar bid phrases in the same cluster are put together. Note that the path for each leaf will not be very long: we have selected the threshold of the clustering algorithm described in Section 4 such that it limits the size of each cluster appropriately.

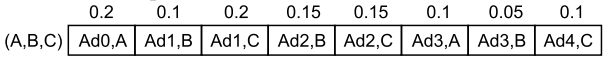
5.2 Block-based Index Structure

H. Bast et al. [4] presented an optimized index structure to support efficient prefix search. The proposed block-based index precomputes unions of inverted lists for groups of words organized according to their lexical closeness. We borrow the main idea of this work and leverage the efficiency of the index structure for efficient ad search. Bid phrases that share a common prefix (i.e. cluster path) are placed together in a same block. Inverted lists of these bid phrases in each block are merged into a list of pairs of bid phrases and their corresponding ads.

Figure 4 compares the traditional inverted index structure with the block-based index structure of AdSearch for three bid phrases A, B and C. Phrase A is bided on ads Ad0 and Ad3, phrase B is bided on Ad1, Ad2 and Ad3, and phrase C is associated with Ad1, Ad2 and Ad4. In the traditional inverted index structure, three separate inverted lists are created. Each list contains a bid phrase as the index term and its associated ads as documents in the posting list. In the block-based index structure, all three bid phrases are placed in a same block together with their corresponding ads because these phrases belong to the same cluster whose ID is 6. Each item in the block is in the form of a pair of an ad and one of its associated bid phrases (e.g. (Ad3, B)), and the block is sorted according to the lexical order of ads. When B is submitted as a user query, the phrases A and C will also be considered as they are similar to B and belong to the same cluster. Thus, Ad1, Ad2 and Ad3 as well as Ad0 and Ad4 are returned as the final result to match the user information need. The score associated with each bid phrase with respect to an ad is calculated by normalized *tf·idf* [2], where *tf* represents the frequency of appearance of a bid phrase in the ad and *idf* indicates the inverse frequency of its occurrence in the whole ad space.



(a) An example of the traditional inverted index structure



(b) An example of the block-based index structure

Figure 4: A comparison of two index structures

In the case of ad search, the proposed block-based index significantly outperforms the traditional inverted index in terms of processing time during query expansion. It has the following advantages: (a) similar bid phrases and their corresponding ads are placed together, which allows sequential scans on a single block or continuous blocks rather than random accesses to separate inverted lists, saving I/O costs; (b) merge operations become fewer or even can be avoided. Expanding phrase B with phrases A and C based on the traditional inverted index structure requires locating the three separate posting lists through random access followed by two merge operations. Based on the block-based index structure, however, the search execution is much more efficient. Only a sequential scan of a single block is required to retrieve ads associated with the related bid phrases, and no merge operations are needed. The detailed algorithm and complexity analysis of efficient query expansion with the block-based index can be found in [4].

The block size will impact the query processing time. A large block usually contains diverse clusters of bid phrases and associated ads, which causes redundant scans on unrelated bid phrase-ad pairs during query expansion. If we use too small blocks, bid phrases belonging to the same cluster will be divided into several blocks. This increases the number of merge operations required to find all relevant ads. H. Bast et al. [4] discussed the selection of a certain block size to balance these effects. However, since clusters of bid phrases correspond to different numbers of ads, adopting a fixed block size does not lead to high efficiency. In our implementation, the size of each block varies according to the number of phrase-ad pairs in each cluster. We do not divide phrase-ad pairs that belong to the same cluster into different blocks. The search performance of indices with fixed and unfixed block sizes will be compared in Section 7.3.

5.3 Dictionaries

A dictionary D is used to record the mapping of each bid phrase to its corresponding artificial words. D also helps locate the corresponding block a bid phrase belongs to. It can be easily constructed by traversing a cluster tree from the root to each bid phrase through the unique path, as described in Section 5.1.

To guarantee that a search engine will return enough ads, an additional dictionary C (i.e. the counter dictionary) is used to record the

number of distinct ads in each cluster. We store the pair in the form of $(c_{p\_} \cdots \_c_{i+1}\_c_i, |ads|)$ for $i \geq 0$ in C where $c_{p\_} \cdots \_c_{i+1}\_c_i$ denotes the cluster path from root c_p to c_i and $|ads|$ is the number of distinct ads which are associated with any of the bid phrases in c_i. Specifically, if c_i represents a bid phrase, $|ads|$ records the number of its associated ads. Note that an artificial word can be converted to the corresponding cluster path by change ":" to "_". During the agglomerative clustering, the amount of distinct ads can be easily obtained by computing the union of the inverted lists of bid phrases in the cluster.

6. QUERY PROCESSING

6.1 Finding Related Bid Phrases with Corresponding Ads

Given a query submitted by a user to a search engine, the process to find related bid phrases with all their corresponding ads is as follows:

1. As input, user queries may contain one or more bid phrases. We look up the dictionary D to get the corresponding artificial words.

2. For each bid phrase, we try to find the minimum cluster in its corresponding cluster tree that contains enough ads, starting from the leaf in a bottom-up manner. This can be achieved efficiently by performing a binary search on top of the counter dictionary C to check whether the current cluster contains no less than m distinct ads. If such a cluster has been found, or if we arrive at the root of the cluster tree, we are ready for the next step. Otherwise, we shorten the cluster path to its half and continue the search with its ancestor cluster.

   Note that we set $m = 1.5 \times k$, ensuring that the exact top-k relevant ads are returned.

3. For each returned cluster, all bid phrases that it contains form the set B. Ads that contain at least one bid phrases in B are stored in the same block, according to the index structure with unfixed block size described in Section 5.2 in detail. Finding all these ads is done through a prefix search on top of the block index, by scanning the block in a sequential order. For the index stucture with fixed block size, bid phrases in B may be divided into several blocks, which also contain pairs from other siblings of the cluster-tree or even other clusters. Thus, we need to filter out irrelevant pairs from these blocks.

4. After collecting pairs of related bid phrases and ads for each input bid phrase, we perform a multi-way merge operation to get the final result.

6.2 Ranking Top-k Relevant Ads

In the above subsection, we presented a procedure to expand the user query with related bid phrases and get a list of ads, each of them associated with at least one of these phrases. In order to find the top-k relevant ads, a scoring function adapted from the spreading activation mechanism is applied to ad ranking, which can improve search precision. Ads that are associated with more bid phrases identical or similar to that of the original query are more likely to meet users' interest and thus to be more relevant. The ranking formula is defined as

$$score(ad) = \sum_{x \in Q} \sum_{y \in B(x)} \sigma_{new}(x, y) \cdot tfidf(y, ad), \qquad (4)$$

where Q refers to the set of query terms, and $B(x)$ is the set of related bid phrases of term x, returned by the process given in Section 6.1. $\sigma_{new}(x, y)$ represents the similarity between x and y. The concept of $tfidf(y, ad)$ has been explained in Section 5.2. The similarity score $\sigma_{new}(x, y)$ is pre-computed and stored in a similarity matrix during the off-line step. It is computed by exploiting the similarity graph generated by the agglomerative clustering to discover latent similarities. In the similarity graph, each edge represents the relatedness between two bid phrases. We iteratively find the maximum similarity value for each pair of vertices in the similarity graph and filter out the edges whose corresponding similarity values are less than 0.05 (the threshold used to terminate the clustering process described in Section 4.2). The whole process is similar to finding shortest paths. For instance, in the similarity graph of Figure 2, $\sigma_{new}(\mathrm{A}, \mathrm{C})$ will finally equal 0.125 (which is $\sigma_{new}(\mathrm{A}, \mathrm{B}) \times \sigma_{new}(\mathrm{B}, \mathrm{C})$).

7. EXPERIMENTAL EVALUATION

7.1 Experiment Setup

In our experiments, both a Chinese dataset and an English dataset are used. As for the Chinese dataset, we first crawled the result pages from a popular Chinese search engine Baidu[1] according to its query logs. From these result pages, we collected ads with their titles and snippets. Due to a lack of bid phrases for the Chinese dataset, we extracted potential bid phrases from the textual content of each ad. These candidates were generated through a use of the n-gram[2] model to enumerate the consecutive two-word, three-word and four-word phrases, and the meaningless ones were further filtered out according to a Chinese dictionary. The English dataset was generated using a similar method by crawling result pages from a commercial Web search engine.

Table 1: Statistical information for both datasets

	Chinese	English
Number of bid phrases	76,946	112,997
Number of ads	524,868	3,008,962

Table 1 shows the statistical information (i.e. the number of distinct bid phrases and the number of distinct ads) for both datasets. The use of datasets in different languages aimed to show our capability to support efficient ad search in a language independent way. We have implemented the block-based index from scratch and developed the traditional inverted index using the open source search API Lucene[3]. Both indices are built on top of the same real datasets mentioned above. As compression algorithms are applied to the block-based index, it occupies no more space than the state-of-the-art optimized inverted index, as shown in Table 2. It conforms to the experimental findings discussed in [4].

Table 2: Index Size (MB)

	Chinese	English
Traditional inverted index	24	161
Block-based index	25.7	163

During query processing, the top-k relevant ads are returned to the users. In our current implementation, k was set to 10. We

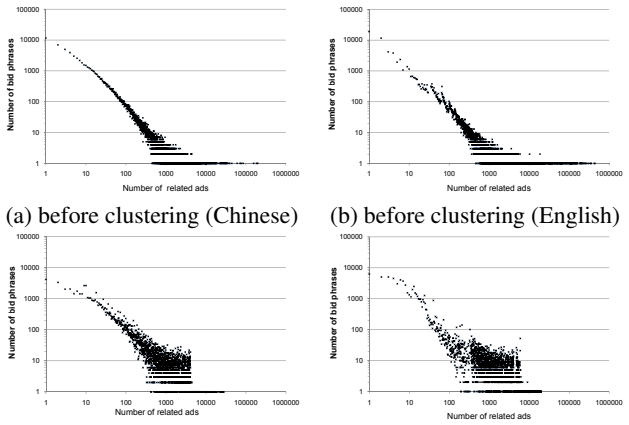
[1] http://www.baidu.com
[2] http://en.wikipedia.org/wiki/N-gram
[3] http://lucene.apache.org/

imposed a constraint that the number of distinct ads contained in the expanded cluster should be no less than 15 unless it arrives at the root of a cluster tree. We submitted the prefix query containing the corresponding cluster path to both indices for query expansion to compare their processing time as well as the quality of search results. Our experiments were conducted on a PC with a 2.6 GHz Pentium processor and 2 GB main memory, operating in 32 bit mode, running Windows.

In order to test our solution in terms of efficiency and effectiveness, we designed the query sets as follows:

- We randomly sampled 100 bid phrases from different domains in each dataset. Each bid phrase is associated with few distinct ads (i.e. the number of associated ads is less than 10). These phrases constituted two query sets referred to as CQS1 and EQS1 for Chinese and English, respectively.
- We selected 100 pairs of sample bid phrases, such that each pair could return ads associated with both bid phrases inside it. These pairs of bid phrases form the query sets referred to as CQS2 and EQS2 for Chinese and English, respectively.
- CQS3 and EQS3 were constructed similarly with queries composed of three or four bid phrases. In this way, the impact of query lengths on processing time was considered.
- We collected 100 popular bid phrases to build the Chinese Frequent Query Set (CQF) and English Frequent Query Set (EQF) for the two respective datasets to evaluate the impact of the frequencies of bid phrases.

7.2 Evaluation of the Clustering Step



(a) before clustering (Chinese) (b) before clustering (English)

(c) after clustering (Chinese) (d) after clustering (English)

Figure 5: Data distribution, the y-axis representing the number of distinct bid phrases and the x-axis representing the number of related ads.

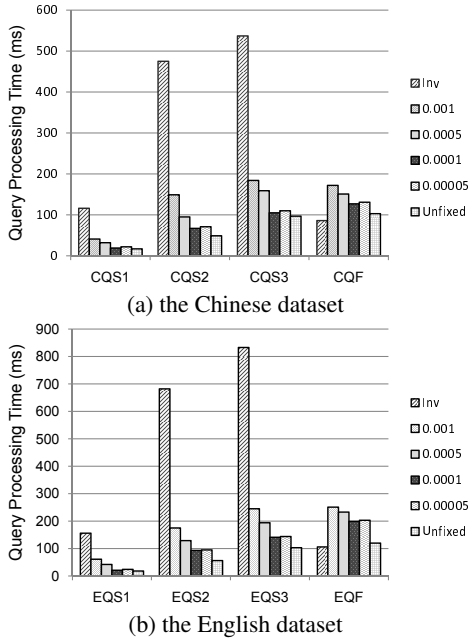
Figure 5 (a) and (b) illustrate the distribution of bid phrases and associated ads in both datasets where both data distributions obey the power-law. It indicates that a large amount of bid phrases are only associated with a rather limited number of ads. Thus, many relevant ads can not be found using purely syntactic approaches because the query does not contain certain terms.

To evaluate the impact of our clustering algorithm on the data distribution, we also recorded the distribution of bid phrases and associated ads after clustering, as shown in Figure 5 (c) and (d). Compared with Figure 5(a) and (b) respectively, the number of bid phrases that only corresponded to few relevant ads was reduced significantly. After clustering, the portion of bid phrases associated with 5 ads or less dropped from 39.7% to 16.7% for the Chinese dataset, and decreased from 35.6% to 21.2% for the English dataset. This meant for both datasets, we moved down a large portion of bid phrases that could not get enough ads. Thus, the clustering process helps a larger number of bid phrases to find enough relevant ads. The effectiveness of clustering are language independent, which indicates that we used the right feature (relationships between ads and bid phrases).

For the Chinese dataset, each cluster contains 9 bid phrases on average, while the largest cluster contains 141 bid phrases. The average depth of clusters for bid phrases is 6 and the maximal depth is 113. For the English dataset, each cluster contains 7 bid phrases on average and the average depth is 7. The largest cluster includes 91 bid phrases and the maximal depth of bid phrases is 79. The distribution of cluster sizes supported the usefulness for us to consider blocks of varying sizes, which will be discussed in the next subsection.

7.3 Efficiency Evaluation

We compared the block-based index with the traditional inverted index in terms of query processing time on the query sets described in Section 7.1. We implemented AdSearch with both fixed-size and unfixed-size blocks to compare the impact of block sizes. In the case of fixed-size blocks, we selected different block sizes and tested their influence on the query processing time. The block-size factor is defined as the fraction of distinct ads in the block with regard to the whole ads. For example, AdSearch (0.001) that the average number of distinct ads contained in each block, for example, the value for the Chinese dataset is about 525 (i.e. 524,868 $\times$ 0.001 $\approx$ 525). Our baseline system Inv is implemented to perform query expansion on top of the traditional inverted index. We repeated the experiments ten times for each query to guarantee the stability of the performance. Each run performs cold start to ensure that index is not cached in memory.



(a) the Chinese dataset

(b) the English dataset

Figure 6: Processing time (ms) for all query sets

Figure 6 (a) and (b) list the average query processing time of both the traditional inverted index and the block-based index of

AdSearch with different block sizes. We found that:

- Block-based index is very useful in the scenarios of efficient ad search. Since a query in CQS1 (EQS1), CQS2 (EQS2) or CQS3 (EQS3) requires to expand related bid phrases for enough relevant ads, AdSearch is much faster than the sytem based on the traditional inverted index. The performance gain is due to the benefits of reducing the merge operations by pre-computing the unions of inverted lists of related bid phrases and the use of sequential scans on these compact blocks instead of random accesses. For queries in CQF and EQF (each of them can return enough relevant ads without query expansion), the performance of the block-based index and the traditional inverted index were almost the same.

- Query processing time increased in direct proportion to the query length. In fact, when considering a query in EQS2 (CQS2) or EQS3 (CQS3), we transformed the original query into several queries in EQS1 (CQS1) and merged ads returned by these new queries. The time needed to merge ads was relatively short compared with the whole query processing time as the query length is usually rather short.

- Query processing time is very sensitive to the block size used in AdSearch. If the block-size factor is too large, it may lead to many unnecessary scans on irrelevant phrases. On the other hand, if the blocks are too small, the search engine will be made access several blocks for each query, increasing the time cost of merge operations. Contrary to the case of Web document retrieval, the number of ads on the Web is limited and the length of each ad is short. Thus, the original block size factor of 0.2 used in [4] cannot be directly applied to ad search. In our experiment, the block size factor varied from 0.001 to 0.00005. in the case of fixed-size blocks, AdSearch (0.0001) had the shortest response time. The query processing time kept decreasing from AdSearch (0.001) to AdSearch (0.0001) for all query sets of both Chinese and English datasets. However, the processing time increased slightly on AdSearch (0.00005) compared with that of AdSearch (0.0001). As AdSearch (0.00005) used too small blocks, the clusters of bid phrases were divided into several blocks so that it required additional merge operations during query expansion. When we applied flexible block sizes, the algorithm gave the best performance on both datasets. This is due to the fact that we kept the cluster in a single block which avoided unnecessary merge operations while still preserving the benefits of sequential scans.

7.4 Effectiveness Evaluation

We use P@n as an effective metric to evaluate the performance in terms of precision. The formula of P@n is defined as

$$\text{P@}n = \frac{|relevant\ ads|}{n} \tag{5}$$

i.e., the portion of relevant ads in the top-n result list. When we set n to a relatively small number, P@n is sensitive to the ranking position of the first one or two ads, and thus helps measure the performance gain on precision of our ranking function. When we set n to a larger number, P@n is sensitive to the amount of relevant ads returned, and thus is helpful to measure the recall improvements benefited from query expansion.

In our experiment, we randomly selected 50 queries from each query set and invited ten people of different backgrounds to manually evaluate the quality of the returned ads. We divided the relevance between an ad and the query containing one or more bid phrases into two levels, which are *relevant* and *irrelevant*. Each query was evaluated by four persons and they were asked to classify the returned ads from both systems (i.e. AdSearch and Baidu) for each query into the above two levels.

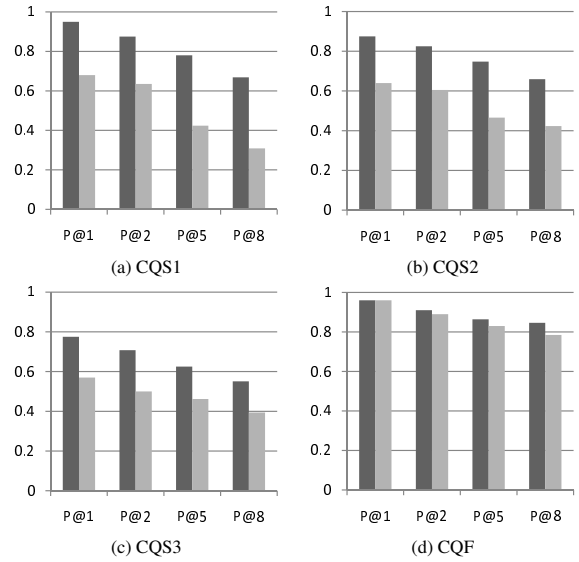


Figure 7: P@n of AdSearch (black) and Baidu (gray) for queries on the Chinese dataset

Figure 7 shows the precisions of AdSearch and Baidu for all four Chinese query sets. In CQF that contains bid phases with high frequency and where query expansion is thus not needed, the performance of both systems was almost the same. However, in the other three query sets, the precision of AdSearch was much higher than that of Baidu. We selected the top 8 results for each query and ran a t-test over all the Chinese query sets on both AdSearch and Baidu, and got a one-tail p-value of 0.000687. This indicated that query expansion could help to retrieve more high-quality ads. Especially for queries which return few results on Baidu, AdSearch can find more relevant ads. Additionally, spreading activation helped AdSearch to beat Baidu as it further considers the latent similarity relationships between bid phrases. In other words, the similarity between bid phrases may help when pursuing a precision oriented ad search.

Table 4: P@n for queries on the English dataset

	EQS1	EQS2	EQS3	EQF
P@1	0.925	0.875	0.735	0.945
P@2	0.850	0.795	0.725	0.868
P@5	0.763	0.722	0.647	0.767
P@8	0.664	0.643	0.556	0.679

Precision for AdSearch on the English dataset is shown in Table 4. The results indicated that query expansion with the ranking scheme did help to find the most relevant ads, especially for the top ones. We plan to give a further study to compare with other baseline algorithms for query expansion in our future work.

7.5 A Query Expansion Case Study

This case study aimed at showing the effects of query expansion on ad search. The expanded bid phrases and the ranked ads are shown in Table 3.

Table 3: Example bid phrases as keyword queries, along with their expanded bid phrases and relevant ads

Bid phrases	Expanded phrases	Top three ads with support of query expansion
"travel agency"	"travel deals" "hotel reservations" "air tickets"	1)Check out travel deals, book air tickets, and much more... 2) Make bookings or enquire about the travel and the hotel reservations services... 3) Check out tour packages, hotels and air tickets. Make an online booking...
"home study courses"	"online universities" "distance learning" "online degrees"	1) Go to top online universities. Get online university info you want... 2) Offers distance learning programs by level, by subject, or by university... 3) Free information on online degrees and distance education college degree programs...
"film videos"	"online mini-series" "teaser trailers" "movie clips"	1) Free downloads for the latest smash movies and online mini-series... 2) Watch new teaser trailers, clips and previews including the latest cinema releases... 3) Browse our movie video library and find great movie clips...
"World Cup"	"football cloth" "Italian Series A" "Europe cup"	1) Football cloth, Number printing, cloth of Italian Series A, World Cup, Europe Cup... 2) Football cloth, Basketball cloth, number printing, World Cup, national team and clubs... 3) Professional flags, German World Cup flag manufacture...

From the table, we can see that all of the expanded bid phrases and their corresponding ads are relevant. Notice that some of the ads could not be retrieved by the original query because they do not contain the keywords. They were retrieved by AdSearch because of the phrases in the same cluster that matched the bid phrases of the queries. Besides, query expansion also affected the ranking of the results. The ads containing a larger number of relevant bid phrases were ranked higher. Take the bid phrase "World Cup" from Table 3 as an example, if we did not perform query expansion, the three ads would be ranked in the order of 3, 1 and 2. Because of the expanded bid phrases, the ranks of 1 and 2 were increased and the final ranking was in the order of 1, 2 and 3. In this way, the ranking scheme helped improve the effectiveness.

8. CONCLUSION

Web advertising has become an important research area and has already attracted a lot of interests from both industrial and academic communities. In this paper, we proposed an efficient ad search solution by extending the existing research on index structures to tackle the issues associated with query expansion, and leverage an effective ranking mechanism adapted from the spreading activation algorithms to get the top-k relevant ads. The experimental results of our prototype system AdSearch showed the efficiency and effectiveness of our approach, regardless of the search query, task domain or the language used.

In the future, we consider to leverage evolutionary clustering algorithms and incremental index update approaches to handle the changes of advertisement data. We also plan to exploit the possibility of using keyword extraction or content mining to uncover latent relationships between bid phrases and ads, which could be integrated into a more advanced relevance matching function.

Acknowledgement

We thank the anonymous reviewers for their valuable and constructive comments. Gui-Rong Xue thanks National Natural Science Foundation of China (NO. 60873211) for such generous support.

9. REFERENCES

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

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. Addison-Wesley Harlow, England, 1999.

[3] H. Bast, D. Majumdar, and I. Weber. Efficient interactive query expansion with complete search. In *CIKM*, 2007.

[4] H. Bast and I. Weber. Type less, find more: fast autocompletion search with a succinct index. In *SIGIR*, 2006.

[5] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD*, 2000.

[6] A. Broder. Computational advertising. In *SODA*, 2008.

[7] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR*, 2007.

[8] A. Z. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *SIGIR*, 2007.

[9] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, 2008.

[10] J. J. Carrasco, D. C. Fain, K. J. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graph. In *ICDM*, 2003.

[11] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *WWW*, 2008.

[12] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *WWW*, 2008.

[13] F. Crestani. Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review*, 11(6):453–482, 1997.

[14] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, 2001.

[15] K. Hosanagar and M. Schwarz. Tutorial on sponsored search. In *EC*, 2007.

[16] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[17] B. J. Jansen. Adversarial information retrieval aspects of sponsored search. In *AIRWeb*, 2006.

[18] A. Lacerda, M. Cristo, M. A. Gonçalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *SIGIR*, 2006.

[19] 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 *SIGIR*, 2008.

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

[21] C. Wang, P. Zhang, R. Choi, and M. D'Eredita. Understanding consumers attitude towards advertising. In *AMCIS*, 2002.

[22] W. V. Zhang, X. He, B. Rey, and R. Jones. Query rewriting using active learning for sponsored search. In *SIGIR*, 2007.