Online Expansion of Rare Queries for Sponsored Search

Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich,
Vanja Josifovski, Donald Metzler, Lance Riedel, Jeffrey Yuan

Yahoo! Research
2821 Mission College Blvd.
Santa Clara, CA 95054
{broder,ciccolo,gabr,vanjaj,metzler,riedell,yuanjef}@yahoo-inc.com

ABSTRACT

Sponsored search systems are tasked with matching queries to relevant advertisements. The current state-of-the-art matching algorithms expand the user's query using a variety of external resources, such as Web search results. While these expansion-based algorithms are highly effective, they are largely inefficient and cannot be applied in real-time. In practice, such algorithms are applied offline to popular queries, with the results of the expensive operations cached for fast access at query time. In this paper, we describe an efficient and effective approach for matching ads against rare queries that were not processed offline. The approach builds an expanded query representation by leveraging offline processing done for related popular queries. Our experimental results show that our approach significantly improves the effectiveness of advertising on rare queries with only a negligible increase in computational cost.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

sponsored search, query expansion, tail queries

1. INTRODUCTION

Search engines provide a gateway to the Web for most Internet users. They also support the Web ecosystem by providing much needed traffic to many Web sites. Each query submitted to a commercial search engine results into two searches. The first search is over the corpus of Web pages crawled by the search engine. The Web crawl performed by the search engine can be viewed as a *pull* mechanism used to obtain documents. The second search is over the corpus of advertisements provided to the search engine through an interface or a feed from advertisers. We can view this as a search over *pushed* content. The ad search provides traffic to (mostly) commercial Web sites that might otherwise not show up in the top Web search results for the query. Since advertisers pay for the placement of their ads on the result page, the search of the ad space is commonly called *sponsored search*. The two main scenarios of sponsored search advertising are *exact match*, where advertisers specify the exact query (bid phrase) for which the ad is to be shown, and *broad match* where queries are matched against ads using a broader criterion. This typically includes matching the query against the ad text, target Web site (landing page), or other information related to the user, ad, or advertiser.

It is a well known fact that the volume distribution of Web search queries follows the power law [28]. The most frequent queries compose the *head* and *torso* of the curve, while the low volume, rarer queries make up the *tail* of the curve. While individually rare, tail queries make up a significant portion of the query volume. For this reason, tail queries have significant potential for advertising revenue.

Web search engines return results for most queries, including those in the tail of the curve. However, this is not the case for sponsored search. Our evaluation of two major search engines has shown that only about 30%-40% of the query volume is covered by ad results. The main reason for this is that tail queries are harder to interpret. In most cases there are no ads that are explicitly associated with them by advertisers who specifically bid on the query. Furthermore, ad matching based on analyzing historical click data is also difficult, since due to the low volume it is harder to accumulate enough ad clicks to use statistical and explore-exploit methods to identify good ads. Search engines normally avoid displaying irrelevant ads in order not to degrade user experience [30], and so the current practice is not to advertise on most of the tail queries.

In this paper we propose a method for online rewriting of tail queries for sponsored search. In our system, we preprocess a large number of head and torso queries offline by expanding them with features extracted from Web search results and store the results in a lookup table. We have shown in our previous work that such expanded queries can be effectively be used to produce query rewrites for broad match [5]. At runtime, we look queries up in the table, and if the query is present, we use the expanded query to search the ad space. While this approach is efficient for head and torso queries, tail queries are too rare and cannot be expanded ahead of time. Expanding them online with Web results would require the sponsored search to wait for the Web search to finish prior to performing ad selection, which in many cases would result in unacceptable latency. To solve this problem, we use the data of the pre-processed queries in a different way. Instead of an exact match lookup, we build

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

an *inverted index* of the expanded query vectors, where each 'document' represents a query and its features. At runtime, when the direct lookup into the query table fails, we use the inverted index to perform a similarity search between the user's query and the pre-processed queries. We then use the features of the top-k most similar queries returned by this procedure to construct an enriched query, which is subsequently used to search over the ad space.

Although our primary focus is sponsored search, our proposed approach is rather general. Indeed, the method can easily be applied to a variety of other search tasks that require expanding rare queries with external knowledge to improve textual matching by overcoming the *vocabulary mismatch* problem. Other potential applications include web search, enterprise search, and social media search.

The primary contributions of this paper are fourfold. First, we present an efficient online query expansion approach for tail queries. Second, we describe a novel approach for directly indexing query expansions for fast computation of query-to-query similarity. Third, we propose a formalism for expanding queries with features derived from pre-expanded related queries. Finally, we describe a ranking and scoring method that adapts standard information retrieval techniques to the structure of the ads by modifying a unit of retrieval.

The remainder of this paper is laid out as follows. We begin with an introduction to sponsored search in Section 2 and describe related work in Section 3. Next, Section 4 explains the general architecture of our system and its various components. In Section 5 we describe our online ad matching algorithm that leverages offline processing. Section 6 details our empirical evaluation over a large sponsored search data set. Finally, Section 7 concludes the paper.

2. BACKGROUND: TEXTUAL ADVERTISING ON THE WEB

Textual ads, which are the ubiquitous short text messages typically marked as "sponsored links", make up a large portion of the Web advertising market. Such ads are commonly distributed through one of the following two advertising channels:

1. *Sponsored Search* or *Paid Search Advertising* places ads on the search result pages of a Web search engine, with ads being targeted to a user's query. All major Web search engines support sponsored search, thereby acting both as a Web search engine and an ad search engine.
2. *Content Match* or *Contextual Advertising* places commercial ads on Web pages. Nearly all of the for-profit non-transactional Web sites rely, at least to some extent, on revenue from contextual advertising. Content match supports a wide range of Web sites, ranging from individual bloggers and small niche communities to large publishers such as major newspapers.

The primary focus of this paper is sponsored search, which is an interplay of the following three entities:

1. **Advertisers** supply the textual ads. The activity of the advertisers is typically organized around *campaigns*, which are a set of ads with a specific temporal and thematic goal (e.g., sale of vacation packages during peak vacation times). The goal of the advertisers, as with traditional advertising, can broadly be defined as promotion of products or services.
2. **Search engines** provide "real estate" for placing ads (i.e., space on search result pages). The goal of the search engine is to select ads that are relevant to the user's query.
3. **Users** interact with the advertisements. Typical interactions include viewing, and potentially, clicking on advertisements. Clicks typically result in the user being redirected to the Web pages of the advertiser.

Sponsored search is usually used as a form of *direct marketing*, as opposed to *brand advertising* that seeks to promote brand awareness in general. Direct marketing advertising aims for a "direct response," where the effect of a campaign can be measured by the user reaction, which may include purchasing advertised goods and services. One of the advantages of online advertising, compared to traditional media, is that it is relatively easy to measure user response. The desired reaction to a sponsored search advertisement is for the user to click on the ad and follow the link to the advertiser's Web site.

When a user clicks an ad, the advertiser pays a certain amount of money. This is known as the pay-per-click or PPC pricing model. Other pricing models exist, including pay-per-impression, where an advertiser pays every time their ad is displayed, and pay-per-action, where advertisers only pay if the ad results in a sale or similar type of transaction. Throughout the remainder of this paper we will assume a PPC model, although the pricing model does not directly effect the usefulness of our underlying model.

The amount paid by the advertiser for each click is typically determined by an auction process [11]. Advertisers place bids on a search phrase, and their position on the search result page is determined by their own bid as well as the bids of other advertisers. Each sponsored search ad has one or more *bid phrases* associated with it. In addition to bid phrases, ads also have a *title* usually displayed in bold font, and an *abstract* or *creative*, which is the few lines of text, usually shorter than 120 characters, displayed on the page. Each ad also contains a URL to the advertised Web page, which is called the *landing page*.

The set of all the ads available in the system can be viewed as structured hierarchically. Each advertiser has one or more *accounts*, which in turn have several *ad campaigns* that usually aggregate ads in the same promotional campaign. Each campaign may have several *ad groups*, which cluster a smaller number of similar ads. Each ad group is composed of an ad *creative*, which is the visible part of the ad displayed to the user, and of one or more bid phrases associated with the ad.

Bid phrases serve two purposes. First, they explicitly specify queries that the ad should be displayed for. Second, they put a price tag on a click event. These price tags could be different for different queries. For example, a contractor advertising his services on the Internet may be willing to pay very little when his ads are clicked from general queries such as "remodeling". However, the contractor may be willing to pay more for focused queries such as "hardwood floors" or "laminate flooring". Ads are most often shown for queries that are listed among the bid phrases for the ad, thus resulting in an *exact match* between the query and the bid phrase. However, it might be difficult, or even impossible, for the advertiser to explicitly list all the relevant queries ahead of

time. For this reason, search engines also have the ability to analyze, and slightly modify queries in an attempt to match the pre-defined bid phrases. This approach, which is called *broad* or *advanced* match, allows more flexible matching of queries to ads, but can be error-prone, which is why not all advertisers opt to use it.

This paper focuses almost exclusively on broad match. Since advertisers tend to be most interested in high volume queries, most bid phrases correspond to head and torso queries. Therefore, most rare queries can only be matched against ads if broad match is employed. Indeed, our proposed algorithm essentially maps rare queries to more common queries and uses information we have gathered offline about the common queries to perform an efficient and effective broad match.

3. RELATED WORK

The techniques presented in this paper are related to the information retrieval research areas of query expansion and query rewriting. The classical work on query expansion by Rocchio [26] shows how to craft the best query for given sets of relevant and irrelevant documents. The most common application of Rocchio's query expansion technique composes an expanded query that is a linear combination of the original query and the relevant/irrelevant documents vectors. Relevant document vectors are added to the query, while irrelevant document vectors are subtracted from the original query.

The literature on query rewriting can be divided into two major directions: 1) incremental rewrites and 2) rewrite into a completely different query. A simple example of the former is stemming, which removes suffixes from query words, so for example, the query "cameras" would match documents with the word "camera" [20, 16, 19]. Other examples of incremental changes include removing one or more query words to improve recall [14], correcting the spelling of words [7] or using a dictionary and thesaurus to find synonyms of query words [29]. Yet another example is pseudo-relevance feedback, which initially retrieves a set of top-ranked documents and uses them for query augmentation [17, 32, 18, 24, 31].

Common methods for completely transforming queries include Latent Semantic Indexing (LSI) [9], which maps all queries and documents to a latent feature space, as well as rewriting queries into related, previously observed queries [3, 8].

Query rewriting is a common technique for performing broad match in sponsored search. Most of the prior work reported in the literature describes offline query rewriting by using various sources of external knowledge, and can thus be applied only to repeating queries from the head and the torso of the query volume curve. In our previous work [21], we proposed query rewriting by using pseudo-relevance feedback to expand ad search queries with features from the algorithmic search results. We used the expanded queries to select an initial set of ads. The bid phrases in the selected ads were considered as candidate rewrites of the original query. A machine learning framework was then used to determine the quality of the candidate rewrites using various lexical and bid-based features. This process was done offline for a large number of queries, and the results were stored in a database for efficient lookup at query time. Using Web search results as a source of external expansion has also been explored by other researchers in the past [10, 27].

Query rewriting for sponsored search reported by Jones et al. [15] employed user session information from search engine query logs to produce alternative queries for ad selection. The first step in this method is to generate candidate substitutions by examining pairs of successive queries issued by the user within the same session. These candidates are then examined to find common transformations. For example, by examining multiple sessions, one can conclude that the word "maps" could be substituted with the word "directions". Thus, when an user issues a query as "new york maps", the system could select ads with "new york directions" as a bid phrase.

To the best of our knowledge, all previous approaches to query expansion for sponsored search used some form of offline processing and simply cached the results for a large number of queries. This allows efficient lookup at query time and tends to result in good ad matching quality. However, the biggest limitation of these approaches is that they cannot handle queries that have not been processed offline ahead of time. For this reason, these approaches are not amenable to handling tail queries, which are very unlikely to have cached results. This results in poor ad matching for tail queries. This is precisely the problem that we address in this paper, namely, how to efficiently and effectively expand tail queries online, in real-time, by leveraging the previous work on offline query processing.

4. SYSTEM ARCHITECTURE

In this section we give a high-level overview of our system, and then the next section presents a more detailed view of the proposed methodology. Figure 1 shows a diagram of the overall system architecture.

The offline processing module pre-computes query expansions for a large number of queries, and then builds an inverted index from the expanded query features. The inverted index maps features of expanded queries into the queries they characterize. In the online phase, when a new query arrives that has not been previously expanded in the offline phase, it is this inverted index that maps the query into related previously-seen queries. This is performed by matching the features of the incoming query against the features of previously expanded queries.

We selected approximately 100 million queries from a Web search log to process offline. The selection of queries was performed based on query volume and how often the query is bidded on in a database of sponsored search advertisements. For each selected query, we expanded it using search results, query logs and click logs as sources of external knowledge. Our approach to query expansion using Web search results is described in [5]. In essence, we retrieve the top-scoring Web search results for each query, extract features from the individual result pages, and then select the most salient features based on the frequency in the result set. In addition to the Web search results, we also use query rewrites generated based on information from query log session as another source of features. The rewrites are selected based on the approach described in [15]. Finally, click log features, which only have a modest impact on the results reported in this paper, are based on proprietary technology whose publication is forthcoming.

At runtime, when a Web query is issued, we first check if it is present in the table of processed queries. If so, we simply retrieve the corresponding expanded query and call the ad

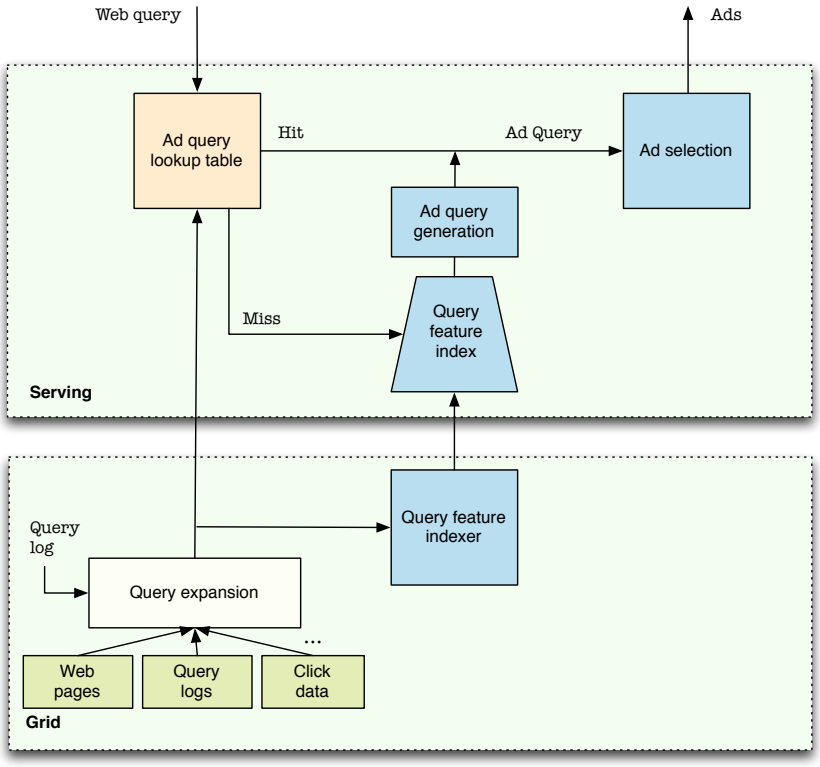

Figure 1: Overview of our system architecture. The bottom box represents the grid cluster that performs all of the offline processing and the top box represents the serving cluster that handles incoming queries.

selection mechanism to retrieve the ads. However, in this paper we focus on the case when the incoming Web query have not been preprocessed beforehand. This corresponds to the cache miss path in Figure 1.

Efficiently matching such unseen queries online is challenging. If we attempt to match just the text of the query against the ad database, then the resulting matches are likely to be of low quality, primarily due to the so-called vocabulary mismatch problem [22]. To alleviate this problem, we first run the incoming query against the inverted index of expanded queries and retrieve the top k expanded queries. We then process the retrieved queries to generate an *ad query*, which is subsequently used to retrieve ads. This results in an expanded version of the original query, even though we have not explicitly pre-processed it offline. This ad query is then passed to the ad selection sub-system to retrieve a set of ads. As we show in Section 6, this computationally-efficient expansion process can significantly improve the ad quality for rare queries. The actual ad selection for rare queries described in this paper is done online in real time, with only a negligible increase in computation compared to running the original, unexpanded query against the ad database.

5. AD MATCHING ALGORITHM

In this section, we describe our proposed approach for matching rare queries to ads in more detail. We focus on several aspects of the query evaluation process. We first describe the features used in the related query selection and examine the scoring mechanisms for this retrieval. Then we describe how to compose the ad query from the results of this retrieval and shortly overview the ad selection.

5.1 Query Feature Extraction

In order to obtain an expressive query representation, we extract three different types of features from each query, including unigrams, phrases, and semantic classes.

For unigrams, terms are stemmed and a small set of stop words are dropped. Phrases are extracted using a phrase dictionary that consists of approximately 10 million statistical phrases gathered from query logs and Web pages [1]. While unigram and phrase features represent a query's syntax, they fail to accurately represent its semantics. Therefore, we also extract a set of semantic classes from each query. This is done by classifying each query into a large hierarchical taxonomy of semantic classes. The taxonomy consists of about 6,000 nodes and has a median depth of 5. We annotate each query with its five most likely semantic classes within the hierarchy. Further details on the classifier can be found in [6].

Table 1 shows an example of the features that may be extracted for the query "low sodium tomato soup recipes". Five unigram features are extracted corresponding to the query terms. Three phrase features corresponding to the phrases "low sodium", "tomato soup", and "soup recipe" are also extracted. Finally, five semantic classes, mostly related to health and cooking are extracted.

Query: low sodium tomato soup recipes		
Unigrams	Phrases	Classes
low	low sodium	health
sodium	tomato soup	health/diet
tomato	soup recipe	health/diet/recipes
soup		cooking
recipe		cooking/soup

Table 1: Example query features extracted from the query "low sodium tomato soup recipes".

5.2 Related Query Retrieval

The unigram, phrase, and class features extracted from the original query act as a pseudo-query made up of features rather than terms. This pseudo-query is then run against our inverted index of queries that have been expanded and pre-processed offline.

We employ a simple, yet effective, vector space-based retrieval approach for retrieving related queries. Within the vector space model, queries and documents are represented as high dimensional vectors. Each vector dimension typically corresponds to a single term, or, in our case, a feature, such as a unigram, phrase, or a semantic class. In our system, features from the original query are weighted as follows:

$$w(f, Q) = (1 + \log \#(f, Q)) \cdot idf(f) \quad (1)$$

where $\#(f, Q)$ is the number of times feature f occurs in query Q and $idf(f)$ is the inverse document frequency for feature f. Here, $idf(f)$ captures the global importance of a feature. It is computed as $idf(f) = \log \frac{N}{N_f}$ where N is the total number of ads in our corpus and N_f is the number of ads that feature f occurs in. Although we compute *idf* based on the ad corpus, it is also possible to compute it based on a query log or a large sample of the Web. Under this weighting, terms that occur in the query and are relatively rare in the ad corpus are given the highest weights, whereas more frequent terms, such as "the" are given considerably lower weights.

The weights for the expanded queries that are stored in the inverted index are computed in a similar manner. However, since the queries in the inverted index are expanded, offline, with Web search results, the weights associated with them have been aggregated over a set of Web search results. In this representation, features are weighted as follows:

$$w(f, E(Q)) = \left(1 + \log \sum_{D \in Results(Q)} \#(f, D)\right) \cdot idf(f) \quad (2)$$

where $E(Q)$ is the Web expanded version of Q, $Results(Q)$ is the set of top Web search results for query Q, $\#(f, D)$ is the number of times that feature f occurs in search result D, and $idf(f)$ is computed based on the ad corpus. In practice, we retrieve the top 40 search results from the Yahoo! search engine and only consider the 50 highest weighted unigrams, 50 highest weighted phrases, and 5 highest weighted classes for each query when building the inverted index. This feature pruning is done to reduce the size of the inverted index and minimize the number of noisy or non-useful features.

For the Web query and expanded query vectors, we adopt the common technique of normalizing the vectors to length 1 (under an l_2 norm). However, rather than normalize vectors across feature types, we only normalize within a given feature types. That is, we ensure that the sub-vector of unigram features has length 1, and similarly for phrase and semantic class features. Although queries are represented as single vectors, they can be conceptualized as three normalized vectors, one corresponding to each feature type.

We define the similarity between two vectors, with respect to a given feature type F, as follows:

$$sim_F(X, Y) = \sum_{f \in F(X) \cap F(Y)} w(f, X) \cdot w(f, Y)$$

where F specifies a feature type and $F(X)$ is the set of features extracted from X of type F. For example, if F_u is the set of unigram features, then $F_u(X)$ is the set of unigram features extracted from X. Thus, $F(X) \cap F(Y)$ is the set of features of type F that occur (i.e., have non-zero weight) in both X and Y. It should be easy to see that $sim_F(X, Y)$ is just the dot product between features of type F in X and features of type F in Y.

In order to produce a final score, the per-feature type similarities are combined via a simple weighted sum. Hence, our scoring function has the following form:

$$sim(Q, E(Q')) = \sum_{F \in \{F_u, F_p, F_c\}} \lambda_F \cdot sim_F(Q, E(Q')) \quad (3)$$

where $E(Q')$ is the Web expanded representation of Q', F_u, F_p, and F_c are the sets of unigram, phrase, and class features, and λ_F signifies the weight associated with each set. Furthermore, in the computation of $sim_F(Q, E(Q'))$, $w(f, Q)$ and $w(f, E(Q'))$ are defined according to Equations 1 and 2, respectively. Thus, our scoring function first computes a dot product for each feature type between the original query and the offline expanded query. The dot products are then combined via a weighted sum (weighted according to λ_F). This formulation provides us the flexibility of assigning different weights to the unigram, phrase, and semantic class feature types, based on the confidence we have in the sources of the external knowledge as the query classifier and the phrase extraction. One issue that can arise from this type of scoring is that the unigram, phrase and class feature vectors could vary in length and thus their normalized components can have different relative impact. For example, as the class vector is of length 3, the components of this vector would have much higher values than the components of the unigram vector that is of length 50 in our experimental setup. We mitigate this vector length difference by taking the vector lengths in account when choosing the λ_F parameters.

Using this ranking algorithm, we retrieve a ranked list of queries that have been processed offline that are related to the incoming (rare) query. As we will now show, these queries can be used to construct an enriched representation of the original query.

5.3 Query Expansion

After the most related queries have been retrieved, we must construct an expanded version of the original rare query, which we will call Q^*. There are many ways to construct Q^*. However, since we are generally working within the vector space model, we use Rocchio's query expansion algorithm, which is known to be effective [26].

Given the original query, represented as a feature vector, and a set of related queries, each also represented as feature vectors, Rocchio's algorithm shifts the original query vector

towards the centroid of the related query vectors. This is a form of pseudo-relevance feedback, where we assume that the related query vectors are relevant and try to push the original query vector in their general direction. This process can be described mathematically as follows:

$$w(f, Q^*) = (1 - \lambda) \cdot w(f, Q) + \lambda \sum_{Q' \in Related(Q)} \frac{w(f, Q')}{|Related(Q)|}$$

where $w(f, Q^*)$ is the weight of feature f in the expanded query vector, $Related(Q)$ is the set of related queries retrieved using Equation 3, $|Related(Q)|$ is the number of related queries retrieved, $w(f, Q')$ is the weight of feature f with respect to Q', and λ is a free parameter that allows us to control the weighting between the original query and the centroid of related queries.

It is important to note the differences between this approach and standard query expansion using pseudo-relevance feedback. First, our approach expands against a small, specialized database of queries, rather than a potentially large, general purpose database (e.g., Web search). As we will show, this can be done very efficiently, unlike querying the Web, which would have to be done offline. Second, rather than expanding using documents directly (query → documents → expanded query), we expand using the search results of related queries (query → related queries → documents → expanded query). This additional level of indirection results in a more diverse set of expansion terms, although it may also result in noisy or spurious expansion features, as well. Since the mapping from related queries to documents has been done offline, the only cost incurred is a lookup, as opposed to the cost of parsing, weight computation, sorting, etc. The end result of the process is an efficient online approximation to standard, inefficient, query expansion approaches.

5.4 Ad Retrieval

We are now ready to explain how ads are scored with respect to the expanded version of a rare query. The approach is similar to how we scored related queries, with a few slight deviations to account for the unique characteristics of ads.

Furthermore, it is important to note that our prototype uses a different ad indexing strategy than what was described in our previous work [5]. In order to overcome the shortness of the ads and allow for more information in the matching process, we use as an entire ad group as a retrieval unit, with all of the bid phrases attached to it. While examining the tradeoffs of this indexing scheme is beyond the scope of this paper, we explain it here since it has a significant impact on how we weight ad features, as we will now discuss.

5.4.1 Ad Feature Weighting

The weighting scheme we used to weight queries is not appropriate for weighting ads. Ads have very different characteristics and must be treated differently. As we just explained, our ad indexing unit supports multiple bid phrases per creative. Advertisements for large advertisers may contain hundreds of bid phrases, while other ads may only have a single bid phrase. Therefore, ad lengths, with respect to the number of unigram, phrase, and class features extracted, have very high variance. Using standard l_2 vector normalization in these cases will cause short ads to be preferred over long ads, which is undesirable. For this reason, we weight ad features using the well-studied BM25 weighting scheme [23, 24], which robustly handles document length normalization and term frequency saturation. The specific form of BM25 weighting that we use is:

$$w(f, A) = \frac{(k+1) \cdot \#(f, A)}{k \cdot \left((1-b) + b \cdot \frac{|A|}{|A|_{avg}}\right) + \#(f, A)} \cdot idf(f)$$

where $|A|$ is the length of the ad, $|A|_{avg}$ is the average ad length, and $\#w(f, A)$ is a weighted count of the number of times that feature f occurs in ad A. Occurrences are weighted according to which section of the ad they occur in, with bid phrase and title occurrences being weighted higher than description and display URL occurrences. This method of combining evidence from multiple fields into a single weighting function has been shown to be effective in the past [25]. In addition, k and b are free parameters that control for term frequency saturation and document length normalization. As before, $idf(f)$ is computed over the entire ad corpus.

5.4.2 Title Match Boosting

Another unique characteristic of ads is their structure. For this reason, we wish to boost the score of ads that have titles that match the query well. To achieve this, we use the following boost factor:

$$prox_F(Q, A) = \frac{\sqrt{\sum_{f \in F(T)} w(f, Q) \cdot w(f, A)}}{\sqrt{\sum_{f \in F(Q)} w(f, Q)^2}}$$

where $F(T)$ are the features of type F extracted from the title. For example, $F_u(T)$ and $F_p(T)$ are the unigram and phrase features extracted from the title. This boost factor acts as a very rudimentary form of term proximity that considers query feature co-occurrences in the title. This serves as a good proximity score approximation, given that no feature position information is stored in our index.

Since we are only interested in matching the text of the titles, we define $prox_{F_c}(Q, A)$, the title match boost for the semantic class features, to be 0.

5.4.3 Scoring Function

Our final ad scoring function is a weighted sum of dot products between features types along with the title match boost. More formally, the scoring function is:

$$S(Q, Q^*, A) = \sum_{F \in \{F_u, F_p, F_c\}} \lambda_F \cdot sim_F(Q^*, A) \cdot (1 + prox_F(Q, A))$$

where Q is the original query and Q^* is the expanded query. Notice that the scoring function takes both the original query and the expanded query as arguments. This is necessary because the title match boost is based on the original query features, not the expanded query features.

Under this scoring function, the best ads are those that have many (highly weighted) features in common with the expanded query $Q*$ and have a title that exactly match the original query Q.

6. EMPIRICAL EVALUATION

This section describes the results of our empirical evaluation of ad matching strategies for tail queries.

6.1 Data

Our offline, pre-processed query lookup table consists of 100 million queries. The set was composed as an union of the top 8 deciles of the queries submitted to the US Yahoo! search engine by volume and the queries that have been used as bid phrases in Yahoo!'s textual ad corpus.

For evaluation purposes, we randomly sampled a set of 400 rare queries from the web search query logs. Of these, 121 were found in the lookup table while the remaining 279 were not. We chose to sample queries in this way so that we could directly evaluate the usefulness of rare queries being found in the lookup table versus our online expansion approach.

For each query, human editors judged the relevance of the top 3 ads returned using several variants of our proposed ad matching algorithms, resulting in a total of 3,556 judgments. Editors labeled the relevance of each ad with respect to a given query on an integral scale from 1 to 5, with a rating of 1 corresponding to a highly attractive ad and and rating of 5 corresponding to a poor ad.

The collection of ads that we matched queries against consists of the entire Yahoo! textual ad corpus.

6.2 Methodology

To evaluate the quality of our proposed ad matching algorithms, we employ discounted cumulative gain (DCG) and precision-recall curves.

The DCG metric is commonly used to evaluate ranking algorithms when there are graded relevance judgments [13]. Since our judgments are on a scale of 1 to 5, this metric is an appropriate choice. The DCG for a single query is defined as as follows:

$$DCG@K(Q) = \sum_{i=1}^{K} \frac{g(i)}{\log(1+i)}$$

where $g(i)$ is the gain associated with the rating of result at rank i and K is maximum depth result to consider. Each gain is discounted, giving less weight to relevant items that appear farther down the ranked list, so the ordering of the results is important. The (arithmetic) average of the per-query DCG@K values, also known as mean DCG@k is used to evaluate an algorithm for an entire set of queries. Since we retrieve three ads per query, we will primarily evaluate our algorithms according to mean DCG@1, DCG@2, and DCG@3. For our experiments, we use gain values of 10, 7, 3, 0.5, and 0 for judgment grades 1, 2, 3, 4, and 5, respectively.

In addition to DCG, we also consider precision-recall curves, which allow us to visualize the tradeoff between recall (fraction of relevant results returned) and precision (fraction of results returned that are relevant). Since precision and recall are defined for binary judgments, we must binarize our judgment grades. For these experiments, we assume that judgment grades 1, 2, and 3 are relevant, while grades 4 and 5 are non-relevant. Furthermore, since we only retrieve three ads per query, plotting classical 11-point interpolated macroaveraged curves makes little sense. Instead, we plot interpolated microaveraged curves [2].

To be rigorous, we often test whether the difference observed between two algorithms is statistically significant or not. We use a paired, one-tailed non-parametric bootstrap test for this purpose [12]. The dagger (†) and double dagger (‡) represent significant improvements with respect to the baseline at the $p < 0.05$ and $p < 0.10$ levels, respectively.

	Baseline	Online Expansion
DCG@1	0.99	1.07 (+8.1%)†
DCG@2	1.57	1.66 (+5.7%)†
DCG@3	1.97	2.10 (+6.6%)‡

Table 2: Comparison of ad matching effectiveness for tail queries not found in the pre-processed lookup table.

6.3 Ad Matching Evaluation

In this section, we evaluate the retrieval quality of our proposed ad matching algorithms. We divide the evaluation into three parts. First, we evaluate queries that are not found in the lookup table of pre-processed queries. Next, we show how effective our method is for queries that are found in the table. Finally, we put it all together and show that we can consistently and significantly improve retrieval quality across the entire spectrum of tail queries by using a hybrid online and offline expansion approach.

We evaluate four different ad matching algorithms, each of which corresponds to a different possible path through the system architecture shown in Figure 1. The first algorithm is Baseline, which ranks results using the original, unexpanded version of the query vector. The second algorithm, Offline Expansion, looks the query up in the lookup table and runs the expanded version of the query, if it exists. Otherwise, it simply uses the original, unexpanded query vector. The third algorithm is Online Expansion and corresponds to our proposed algorithm for expanding queries online using offline processing. The details of the algorithm were described in detail in Section 5. For this algorithm, we expand the original query with 3 related queries from the inverted index. Finally, we refer to the last algorithm as Online+Offline Expansion. As the name suggests, it is a combination of the Online Expansion and Offline Expansion approaches. The algorithm forms an ad vector from a weighted combination of the Online Expansion ad vector and the Offline Expansion ad vector. In our experiments using this algorithm, we give the Online Expansion ad vector a weight of 1 and the Offline Expansion ad vector a weight of 2.

In all of our experiments, we set λ_{F_u}, λ_{F_p}, and λ_{F_c}, the unigram, phrase, and class feature weights, to 1, 1, and 0.1, respectively. These weights were found to produce good results in previous evaluations. Furthermore, the b, k, and field weight parameters used for BM25 weighting are set to previously tuned values that were shown to be effective for sponsored search retrieval.

6.3.1 Queries not Found in Lookup Table

Our primary evaluation concerns those queries that are not found in the lookup table of offline processed queries. These are the queries that we would like to expand, online, to improve the quality of ad matches. For this purpose, we compare the effectiveness of the Baseline and Online Expansion algorithms. We believe our baseline is reasonable and appropriate because the queries that are not found in the lookup table are very rare and there is little that can be done using existing query expansion techniques on these queries, especially online, in real-time.

The results of this evaluation are given in Table 2 and Figure 2. First, the DCG results in Table 2 show that our Online Expansion algorithm consistently and significantly

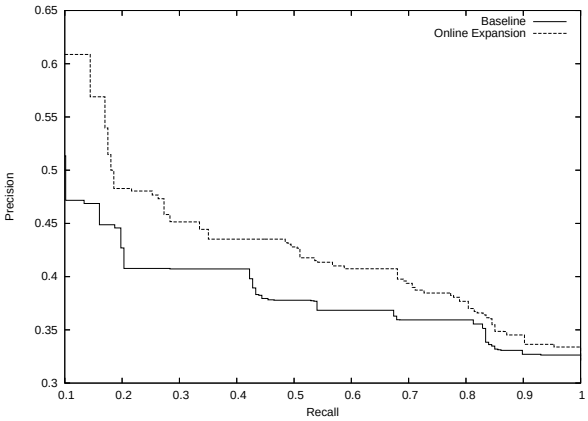

Figure 2: Interpolated precision-recall curve for tail queries not found in the pre-processed lookup table.

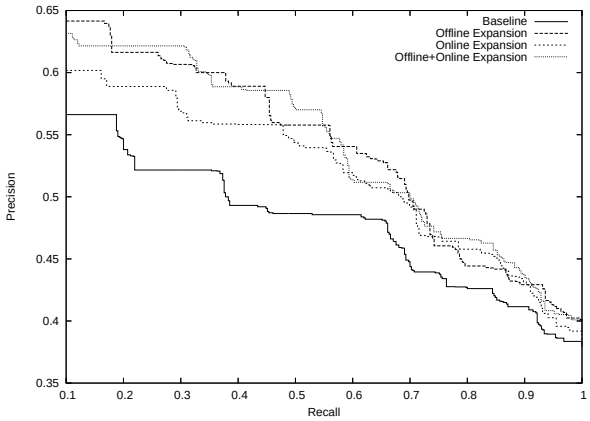

Figure 3: Interpolated precision-recall curve for tail queries found in the pre-processed lookup table.

	DCG@1	DCG@2	DCG@3
Baseline	2.89	4.56	5.75
Online Expansion	2.83	4.43	5.54
Offline Expansion	3.07‡	4.75	5.87
Online+Offline Expansion	2.91	4.44	5.59

Table 3: Comparison of ad matching effectiveness for tail queries found in the pre-processed lookup table.

	DCG@1	DCG@2	DCG@3
Baseline	1.61	2.58	3.23
Online Expansion	1.71‡	2.68	3.32
Offline Expansion	1.66	2.63	3.25
Online+Offline Expansion	1.76‡	2.69	3.37
Hybrid	1.79†	2.78†	3.43†

Table 4: Comparison of ad matching effectiveness for all tail queries.

improves DCG@1, DCG@2, and DCG@3. Indeed, the Online Expansion improves DCG@1 by over 8%. The precision-recall curves in Figure 2 show similar results, with the Online Expansion curve dominating the Baseline curve at all levels of recall. This is an important characteristic, because practical systems often aim for recall levels well below 100% in order to produce higher quality ads. Our results show that our Online Expansion technique is more effective than the Baseline regardless of the level of recall, making it very useful from a practical point of view. These results clearly show that we are able to effectively and efficiently enrich tail queries by leveraging large-scale offline processing, which was the primary goal of this paper.

It is important to note that the absolute DCG values for these queries is somewhat low, primarily due to the very nature of the queries under consideration. As we show in Section 6.4, many of these queries contain misspellings, URLs, and proper names. It is typically difficult to find relevant ads for such queries. It may be possible to use automatic classification techniques, such as the one recently proposed by Broder et al. [4], to determine the quality of a given set of ads. Such methods can be used to improve overall ad quality and reduce the number of times ads are shown for queries with embarrassingly low DCG.

6.3.2 *Queries Found in Lookup Table*

Next, we evaluate the effectiveness of the Baseline, Online Expansion, Offline Expansion, and Online+Offline Expansion approaches on the set of tail queries that are found in the lookup table. The results of our evaluation are shown in Table 3 and Figure 3.

The first thing to notice is that the Offline Expansion approach is consistently better than the other approaches in terms of DCG. This result is not unexpected, however, since we expect offline expansion to be superior to online expansion. Our proposed online expansion approach is really a last resort algorithm that should only be applied to queries that are not found in the lookup table. Thus, since online expansion is really just an approximation for the offline expansion, we expect it to produce better results for queries in the lookup table. The results also show that the combined method, Online+Offline Expansion, is slightly worse than Offline Expansion, but the difference is not statistically significant. Finally, it is interesting to note that the Offline Expansion approach is only significantly better than the Baseline in terms of DCG@1. This is likely due to the fact that the Offline Expansion approach was largely tuned for head and torso queries, and since our evaluation is only done over tail queries, the approach ends up being consistently better than the baseline, but not always significantly better.

The precision-recall curve in Figure 3 clearly shows that the Baseline and Online Expansion approaches are inferior to the Online+Offline Expansion and Offline Expansion approaches. Despite the DCG results, the precision-recall curves suggest that the difference between the Online Expansion and Online+Offline Expansion approaches is not very large.

Therefore, our results for tail queries found in the lookup table suggest that using Offline Expansion is the best strategy and that the Online+Offline Expansion approach is also a valid option.

6.3.3 *Putting It All Together*

We now describe how to put together a highly effective, efficient sponsored search ad matching algorithm for tail queries. Our results up until this point have suggested the

Characteristic	Percentage
Misspelled	21%
Domain/URL	18%
Proper name	14%
Foreign	10%

Table 5: Characteristics of rare query.

Online Expansion approach works the best for tail queries not found in the lookup table and the Offline Expansion approach is the most effective for the tail queries that do happen to appear in the lookup table. Given this, we propose the Hybrid approach that combines the Online Expansion and Offline Expansion methods. The Hybrid approach is very simple, yet, as we will show, very effective, as well. Under the approach, queries that are found in the lookup table are processed using the Offline Expansion method, whereas queries that are not found in the lookup table are processed using the Online Expansion method. Since both of these approaches can be done online, the Hybrid method can also be implemented very efficiently. The underlying rationale behind this approach is to combine the best ranking approaches for both of the query types into a superior ranking function.

We evaluate the effectiveness of the Baseline, Online Expansion, Offline Expansion, Online+Offline Expansion, and Hybrid approaches across the *entire* set of tail queries in Table 4. The results show that the Online Expansion and Online+Offline Expansion approaches are significantly better than the Baseline according to DCG@1. However, the clearly superior approach for handling all tail queries is the Hybrid approach, which improves over the Baseline by 11.2%, 7.8%, and 6.2% in terms of DCG@1, DCG@2, and DCG@3, respectively. The improvements of the Hybrid approach over the Online+Offline Expansion approach for the same metrics are 1.7%, 3.3% (‡), and 1.8%, respectively. Thus, even though our goal was to develop an effective ad matching algorithm for tail queries not found in our lookup table, our proposed Hybrid approach shows consistent and significant improvements in DCG across the entire spectrum of tail queries.

6.4 Characteristics of Rare Queries

To develop a better understanding of rare queries and to help us improve the performance of the system, we analyzed a large set of rare queries to find out the most common cause of mismatched ads. Table 6.4 lists the most common classes we observed. Over a fifth of the tail queries contained at least one misspelled word.

Although the queries were selected from US logs, there is still around 10% of foreign queries. We eliminated these queries from our evaluation. Another common type of a rare query are URLs, as for example when the user types "sendpictureshome.com". Such queries can be processed by parsing the URL and extracting features. However, as these were not the focus of our evaluation, we treat the URL as a single feature. We also noted that a significant portion (14%) of the rare queries contained proper names of people, places or organizations.

6.5 Efficiency

One of our primary claims throughout this paper is that our proposed online query expansion method is computationally efficiently. The table lookup can be implemented very efficiently (approximately 1 ms), resulting in negligible overhead for every incoming query. The more expensive code path is exercised when the incoming query is not found in the lookup table. In this case, a query must be run against the inverted index of expanded queries. The efficiency of this step largely depends on the underlying search infrastructures. Finally, regardless of whether the query was found in the lookup table or not, an expanded query must be run against the database of ads. The efficiency of this, again, depends on the efficiency of the search infrastructure.

Experiments using our prototype system revealed that online query expansion adds approximately 50% overhead, on average, to the query execution time. Assuming that our lookup table covers approximately 75% of the query volume, this results in an overhead of just 12.5% when averaged over the entire query stream.

Therefore, the amount of overhead can be minimized by improving query execution time against the inverted index of expanded queries, increasing the computational power, or increasing the number of queries stored in the initial lookup table. These three factors can be balanced to trade off between effectiveness and efficiency.

7. CONCLUSIONS

This paper has focused on efficient and effective methods for matching ads to tail queries. We proposed an online query expansion approach that leverages large-scale offline processing of queries. In our approach, we build a large inverted index of queries, and their expanded ad vectors, that we run incoming tail queries against. We then use the results to enrich the representation of the original tail query

Our evaluation shows consistent and significant improvements in retrieval effectiveness can be achieved using our proposed approach for tail queries that we have not done pre-processing for, both in terms of DCG and precision-recall analysis. Furthermore, we show that an approach that combines online and offline processing can be effective for tail queries that we have some offline processing information for. Finally, we proposed a hybrid ranking strategy that leverages the best of online and offline query expansion. The hybrid approach shows significant improvements in ad matching quality over the entire spectrum of tail queries.

Acknowledgments

We thank Ann Hsieh and her editorial team for judging the ad relevance under a very tight schedule. We also thank the anonymous reviewers for their comments and suggestions.

8. REFERENCES

[1] P. Anick. Using terminological feedback for web search refinement: a log-based study. In *Proc. 26th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 88–95, 2003.

[2] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[3] B. Billerbeck, F. Scholer, H. Williams, and J. Zobel. Query expansion using associated queries. In *Proc. 12th Intl. Conf. on Information and Knowledge Management*, pages 2–9, 2003.

[4] A. Broder, M. Ciaramita, M. Fontoura, E. Gabrilovich, V. Josifovski, D. Metzler, V. Murdock, and V. Plachouras. To swing or not to swing: Learning when (not) to advertise. In *Proc 17th. Intl. Conf. on Information and Knowledge Management*, pages 1003–1012, 2008.

[5] A. Broder, P. Ciccolo, M. Fontoura, E. Gabrilovich, V. Josifovski, and L. Riedel. Search advertising using web relevance feedback. In *Proc 17th. Intl. Conf. on Information and Knowledge Management*, pages 1013–1022, 2008.

[6] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proc 30th . Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 559–566, 2007.

[7] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 293–300, 2004.

[8] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proc 11th Intl. Conf. on World Wide Web*, pages 325–332, 2002.

[9] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[10] F. Diaz and D. Metzler. Improving the estimation of relevance models using large external corpora. In *Proc. 29th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 154–161, 2006.

[11] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[12] B. Efron and R. J. Tibshirani. *An introduction to the Bootstrap.*, volume 57 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, 1993.

[13] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[14] R. Jones and D. C. Fain. Query word deletion prediction. In *Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 435–436, 2003.

[15] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. 15th Intl. Conf. on World Wide Web*, pages 387–396, 2006.

[16] R. Krovetz. Viewing morphology as an inference process. In *Proc 16th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 191–202, 1993.

[17] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 120–127, 2001.

[18] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 206–214, 1998.

[19] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *Proc. 30th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 639–646, 2007.

[20] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[21] 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 *Proc. 31st Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 403–410, 2008.

[22] B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. S. de Moura. Impedance coupling in content-targeted advertising. In *SIGIR'05*, 2005.

[23] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proc. 17th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 232–241, 1994.

[24] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. 3rd Text REtrieval Conference*, pages 109–126, 1994.

[25] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proc. 13th Intl. Conf. on Information and Knowledge Management*, pages 42–49, 2004.

[26] J. J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. Prentice-Hall, 1971.

[27] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, 2006.

[28] A. Spink, D. Wolfram, B. Jansen, and T. Saracevic. Searching the web: The public and their queries. *Journal of the American Society for Information Science and Technology*, 53(3):226–234, 2001.

[29] E. M. Voorhees. Query expansion using lexical-semantic relations. In *Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 61–69, 1994.

[30] C. Wang, P. Zhang, R. Choi, and M. D. Eredita. Understanding consumers attitude toward advertising. In *Proceedings of Americas Conference on Information Systems*, pages 1143–1148, 2002.

[31] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Science (TOIS)*, 18(1):79–112, 2000.

[32] C. Zhai and J. D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Intl. Conf. on Information and Knowledge Management*, pages 403–410, 2001.