Classifying Search Queries Using the Web as a Source of Knowledge

EVGENIY GABRILOVICH and ANDREI BRODER
Yahoo Research
MARCUS FONTOURA
PUC-Rio
AMRUTA JOSHI
UCLA
VANJA JOSIFOVSKI and LANCE RIEDEL
Yahoo Research
and
TONG ZHANG
Rutgers University

We propose a methodology for building a robust query classification system that can identify thousands of query classes, while dealing in real time with the query volume of a commercial Web search engine. We use a pseudo relevance feedback technique: given a query, we determine its topic by classifying the Web search results retrieved by the query. Motivated by the needs of search advertising, we primarily focus on rare queries, which are the hardest from the point of view of machine learning, yet in aggregate account for a considerable fraction of search engine traffic. Empirical evaluation confirms that our methodology yields a considerably higher classification accuracy than previously reported. We believe that the proposed methodology will lead to better matching of online ads to rare queries and overall to a better user experience.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering; Query formulation; Relevance feedback; Search process*

General Terms: Algorithms, Measurement, Performance, Experimentation

A preliminary version of this article appeared in *Proceedings of the 30th ACM International Conference on Research and Development in Information Retrieval (SIGR)* [Broder et al. 2007].
T. Zhang's research has been partially supported by NSF grant DMS-0706805.
Authors' addresses: E. Gabrilovich, A. Broder, V. Josifovski, and L. Riedel, Yahoo! Research, 2821 Mission College Blvd., Santa Clara, CA, 95054; email: {gabr, broder, vanjaj, riedell}@yahoo-inc.com; M. Fontoura, Computer Science Department, PUC-Rio, Rio de Janeiro, Brazil; email: mfontoura@inf.puc-rio.br; A. Joshi, UCLA Computer Science Department, 4732 Boelter Hall, Los Angeles, CA, 90095; email: amrutaj@cs.ucla.info; T. Zhang, Department of Statistics, Rutgers University, 501 Hill Center, Busch Campus, Piscataway, NJ 08854; email: tongz@rci.rutgers.edu.

DOI 10.1145/1513876.1513877 http://doi.acm.org/10.1145/1513876.1513877

Additional Key Words and Phrases: Pseudo relevance feedback, query classification, Web search

ACM Reference Format:

Gabrilovich, E., Broder, A., Fontoura, M., Joshi, A., Josifovski, V., Riedel, L., and Zhang, T. 2009. Classifying search queries using the Web as a source of knowledge. ACM Trans. Web 3, 2, Article 5 (April 2009), 28 pages. DOI = 10.1145/1513876.1513877
http://doi.acm.org/10.1145/1513876.1513877.

1. INTRODUCTION

In its 15 year lifetime, Web search has grown tremendously; it has simultaneously become a factor in the daily life of maybe a billion people, and at the same time a twenty billion dollar industry fueled by Web advertising. One thing, however, has remained constant: people use very short queries. Various studies estimate the average length of a search query between 2.4 and 2.7 words, which by all accounts can carry only a small amount of information. Commercial search engines do a remarkably good job of interpreting these short strings, but they are not (yet!) omniscient. Therefore, using additional external knowledge to augment the queries can go a long way in improving the search results and the user experience.

At the same time, better understanding of query meaning has the potential of boosting the economic underpinnings of Web search, namely, online advertising, via the *sponsored search* mechanism that places relevant advertisements alongside search results. For instance, knowing that the query “SD450” is about cameras while “nc4200” is about laptops can obviously lead to more focused advertisements even if no advertiser has specifically bid on these particular queries. Advertising lies at the heart of modern Web search monetization, and better understanding of search queries is likely to result in more focused ads, eventually leading to better user experience. Better leveraging of the ads mechanism can also lead search engines to provide additional services to the searchers.

In this study we present a methodology for *query classification*, where our aim is to classify queries onto a commercial taxonomy of Web queries with approximately 6000 nodes. Given such classifications, one can directly use them to provide better search results as well as more focused ads. The problem of query classification is extremely difficult owing to the brevity of queries. Observe, however, that in many cases a human looking at the search query and *the search results* does remarkably well in making sense of it. For instance, in our example, sending the query “SD450” to a Web search engine brings pages about Canon cameras, while “nc4200” brings pages about Compaq laptops, hence to a human the intent is quite clear. Of course, the sheer volume of search queries does not lend itself to human supervision, and therefore we need alternate sources of knowledge about the world.

Search engines index colossal amounts of information, and as such can be viewed as large repositories of knowledge. Following the heuristic we described, we propose to use the search results themselves to gain additional insights for query interpretation. To this end, we employ the pseudo-relevance feedback paradigm, and assume the top search results to be relevant to the query. Certainly, not all results are equally relevant, and thus we use elaborate *voting schemes* in order to obtain reliable knowledge about the query. For the purpose of this study we first dispatch the given query to a general Web search engine, and collect a number of the highest-scoring URLs. We crawl the Web pages pointed to by these URLs, and classify these pages. Finally, we use these result-page classifications to classify the original query. Our empirical evaluation confirms that using Web search results in this manner yields substantial improvements in the accuracy of query classification.

Note that in a practical implementation of our methodology within a commercial search engine, all indexed pages can be preclassified using the normal text-processing and indexing pipeline. Thus, at runtime we only need to run the voting procedure, without doing any crawling or classification. This additional overhead is minimal, and therefore the use of search results to improve query classification is entirely feasible at runtime.

Another important aspect of our work lies in the choice of queries. The volume of queries in today's search engines follows the familiar power law, where a few queries appear very often while most queries appear only a few times. While individual queries in this long tail are rare, together they account for a considerable mass of all searches. Furthermore, the aggregate volume of such queries provides a substantial opportunity for income through online advertising.[1]

For frequent queries, searching and advertising platforms can be trained to provide good results, including auxiliary data such as maps, shortcuts to related structured information, successful ads, and so on. "Tail" queries, however, simply do not have enough occurrences to allow statistical learning on a per-query basis. Therefore, we need to aggregate such queries in some way, and to reason at the level of aggregated query clusters. A natural choice for such aggregation is to classify the queries into a topical taxonomy. Knowing which taxonomy nodes are most relevant to the given query will help us to provide the same type of support for rare queries as for frequent queries. Consequently, in this work we focus on the classification of rare queries, whose correct classification is likely to be particularly beneficial.

The main contributions of this article are as follows. First, we build the query classifier directly for the target taxonomy, instead of using a secondary auxiliary structure; this greatly simplifies taxonomy maintenance and development. The taxonomy used in this work is two orders of magnitude larger than that used in prior studies. Empirical evaluation demonstrates that our methodology for using external knowledge achieves greater improvements than those previously reported. Since our taxonomy is considerably larger, the classification problem we face is much more difficult, making the improvements we achieve particularly notable. We also report the results of a thorough empirical study of different voting schemes and different depths of knowledge

[1] In our examples, "SD450" and "nc4200" represent gadget models that existed on the market for a long time, and hence there are advertisers placing ads on these queries. However, in this article we mainly deal with rare queries that are extremely difficult to match to relevant ads because normally no advertisers specifically bid on those queries.

(e.g., using search summaries vs. entire crawled pages). We found that using the full text of search hits yields deeper knowledge and leads to greater improvements than mere summaries. This result contrasts with prior findings in query classification [Shen et al. 2006b], but is supported by research in mainstream text classification [Gabrilovich and Markovitch 2007].

This article is organized as follows. Section 2 surveys related work. Section 3 provides background on sponsored search advertising. We present our methodology for query classification using Web search in Section 4. The results of empirical evaluation are reported in Section 5. Finally, we discuss the obtained results in Section 6.

2. RELATED WORK

In its abstract form, query classification can be regarded as a multiclass categorization problem, which has been extensively studied in the machine learning literature. If we regard each query as a short text segment, then text categorization techniques can be easily applied. The standard feature representation in text categorization is the so-called *bag-of-word* representation, where the features are the word counts in the text document. For general text categorization problems, this simple representation often achieves state-of-the-art performance. Standard text categorization methods that can be used with this feature representation include nearest neighbor [Yang 1999], naive Bayes [McCallum and Nigam 1998], SVM [Joachims 1998], and more generally, regularized linear classification methods [Zhang and Oles 2001]. See Sebastiani [2002] for a comprehensive survey of text categorization techniques.

In text categorization, each category is often associated with a number of words that are indicative of the category. Since text documents often contain at least a few hundred words, a number of indicative words will likely appear in each document. It is thus relatively easy for a standard machine learning method to find most of these words even with a small amount of training data. A weighted average of the words will give a good estimate of whether a document belongs to a certain category or not. The situation changes dramatically for query classification, where each query often contains only a very small number of words. It is therefore very difficult for standard machine learning methods to find many indicative words for a category from the training queries. In fact, many words that appear in the test queries do not occur in the training queries at all—this problem is also referred to as the *data-sparsity problem* in natural language processing. For this reason, the simple bag-of-word representation for text categorization does not work well for query classification.

Even though the average length of search queries is steadily increasing over time, a typical query is still shorter than three words. Consequently, many researchers have studied possible ways to enhance queries with additional information.

One important direction in enhancing queries is through query expansion. This can be done either using electronic dictionaries and thesauri [Voorhees 1994], or via relevance feedback techniques [Manning et al. 2008] that make use of a few top-scoring search results. Early work in information retrieval concentrated on manually reviewing the returned results [Salton and Buckley 1990; Rocchio 1971]. However, the sheer volume of queries nowadays does not lend itself to manual supervision, hence subsequent works focused on *pseudo* relevance feedback, which basically assumes top returned results to be relevant [Xu and Croft 2000; Mitra et al. 1998; Efthimiadis and Biron 1994; Robertson et al. 1995].

More recently, studies in query augmentation focused on classification of queries, assuming such classifications to be beneficial for more focused query interpretation. Indeed, Kowalczyk et al. [2004] found that using query classes improved the performance of document retrieval.

Studies in the field pursue different approaches for obtaining additional information about the queries. Beitzel et al. [2005b] used semi-supervised learning as well as unlabeled data [Beitzel et al. 2005a, 2007]. Gravano et al. [2003] classified queries with respect to geographic locality in order to determine whether their intent is local or global.

The 2005 KDD Cup on Web query classification inspired yet another line of research, which focused on enriching queries using Web search engines and directories [Li et al. 2005; Shen et al. 2005, 2006a; Kardkovacs et al. 2005; Vogel et al. 2005]. The KDD task specification provided a small taxonomy (67 nodes) along with a set of labeled queries, and posed a challenge to use this training data to build a query classifier. Several teams used the Web to enrich the queries and provide more context for classification. The main research questions of this approach are, (1) how to build a document classifier, (2) how to translate its classifications into the target taxonomy, and (3) how to determine the query class based on document classifications.

The winning solution of the KDD Cup [Shen et al. 2005] proposed using an ensemble of classifiers in conjunction with searching multiple search engines. To address issue (1), their solution used the Open Directory Project (ODP) to produce an ODP-based document classifier. The ODP hierarchy was then mapped into the target taxonomy using word matches at individual nodes. A document classifier was built for the target taxonomy by using the pages in the ODP taxonomy that appear in the nodes mapped to the particular target node. Thus, Web documents were first classified with respect to the ODP hierarchy, and their classifications were subsequently mapped to the target taxonomy for query classification.

Our approach differed in that we solved the problem of document classifications directly in the target taxonomy by using the queries to produce document classifiers, as described in Section 4. This simplifies the process and removes the need for mapping between taxonomies. This also streamlines taxonomy maintenance and development. Using this approach, we were able to achieve good performance in a very large scale taxonomy. We also evaluated a few alternatives for how to combine individual document classifications when actually classifying the query.

In a follow-up paper Shen et al. [2006b], proposed a framework for query classification based on bridging between two taxonomies. In this approach, the problem of not having a document classifier for Web results is solved by using a training set available for documents with a different taxonomy. For this, an intermediate taxonomy with a training set (ODP) is used. Then several schemes are tried that establish a correspondence between the taxonomies or allow for mapping of the training set from the intermediate taxonomy to the target taxonomy. As opposed to this, we built a document classifier for the target taxonomy directly, without using documents from an intermediate taxonomy. While we were not able to directly compare the results due to the use of different taxonomies (we used a much larger taxonomy), our precision and recall results are consistently higher even over the hardest query set.

There have also been a number of studies that studied query classification as a means for accomplishing other tasks. Beitzel et al. [2004] used query classification techniques to analyze a large-scale log of real-life Web queries. Sahami et al. [2004] explored classification of Web queries onto the nodes of the Open Directory Project (`www.dmoz.org`), with the aim of eventually using the query classification capabilities for improving the accuracy of Web search. Lu et al. [2006] used machine learning methods to identify navigational queries (where the users' information need is to find a URL / home page for the entity described in the query).

3. DIGRESSION: THE BASICS OF SPONSORED SEARCH

This research has been motivated by the need to match Web search queries to more relevant ads. Therefore, in this section we provide a brief introduction to some basic concepts of Web advertising. *Sponsored search* (or *paid search*) advertising is placing textual ads on the result pages of Web search engines, with ads being driven by the originating query. All major search engines (Google, Yahoo!, and MSN) support such ads and act simultaneously as a search engine and an ad agency. Sponsored search is an interplay of three players:

- The *advertiser* provides the supply of ads. Usually the activity of the advertisers are organized around *campaigns*, which are defined by a set of ads with a particular temporal and thematic goal (e.g., sale of digital cameras during the holiday season). As in traditional advertising, the goal of the advertisers can be broadly defined as the promotion of products or services.
- The *search engine / ad agency* is a mediator between the advertiser and the user, selecting the ads that are put on the search result pages.
- *Users* visit the search engine to perform queries and interact with the ads.

Sponsored search usually falls into the category of *direct marketing* (as opposed to *brand advertising*), that is advertising whose aim is a direct response, where the effect of a campaign is measured by the user reaction. One of the advantages of online advertising in general, and contextual advertising in particular is that, compared to the traditional media, it is relatively easy to measure the user response. Usually the desired immediate reaction is for the user to follow the link in the ad and visit the advertiser's Web site and the prevalent financial model is that the advertiser pays a certain amount for every click on the advertisement (PPC).

In most networks, the amount paid by the advertiser for each click is determined by an auction process where the advertisers place bids on a search phrase, and their position in the tower of ads displayed in conjunction with the result is determined by their bid. Thus each ad is annotated with one or more *bid phrases*. The bid phrase is a concise description of the target ad audience as determined by the advertiser. In addition to the bid phrase, an ad is also characterized by a *title* usually displayed in a bold font, and an *abstract* or *creative*, which is the few lines of text, usually fewer than 120 characters.

In general, clicks bring benefits to the ad agency by providing revenue, and to the advertiser by bringing traffic to the target Web site. The revenue of the ad agency, given a query q, can be estimated as:

$$R = \sum_{i=1..k} P(click|q, a_i) price(a_i, i),$$

where k is the number of ads displayed on the result page for query p, and $price(a_i, i)$ is the click-price of the current ad a_i, at position i. The price in this model depends on the set of ads presented on the search result page. A discussion of bidding and placement mechanisms is beyond the scope of this article [Moran and Hunt 2005].

It is important to note that many searches do not explicitly use phrases that someone bids on. Consequently, advertisers also buy *broad matches*, that is, they pay to place their advertisements on queries that constitute some modification of the desired bid phrase. In broad match, several syntactic modifications can be applied to the query to match it to the bid phrase, for example, dropping or adding words, synonym substitution, and so forth. These transformations are based on rules and dictionaries. As advertisers tend to cover high-volume and high-revenue queries, broad-match queries fall into the tail of the distribution with respect to both volume and revenue.

4. METHODOLOGY

Our methodology has two main phases. In the first phase, we construct a *document classifier* for classifying search results into the same taxonomy into which queries are to be classified. In the second phase, we develop a *query classifier*, which invokes the document classifier on search results, and uses the latter to perform query classification.

Figure 1 gives an overview of our proposed query classification architecture. This architecture has two distinct components:

—The *Offline* component is invoked once, for all future query classification tasks. This step is performed at crawling time while building the index of a search engine. In this phase, every page that is about to be added to the search engine index is first classified. This way, when we later classify a query based on its search results (see Section 4.2), the classifications of all individual search results are already available and need not be computed on the fly.

Having all indexed pages preclassified offers substantial time savings at runtime. However, a search engine is already an extremely complex system, so that extending it to perform Web page classification at crawling time is a difficult engineering task. Consequently, for the experiments described in this

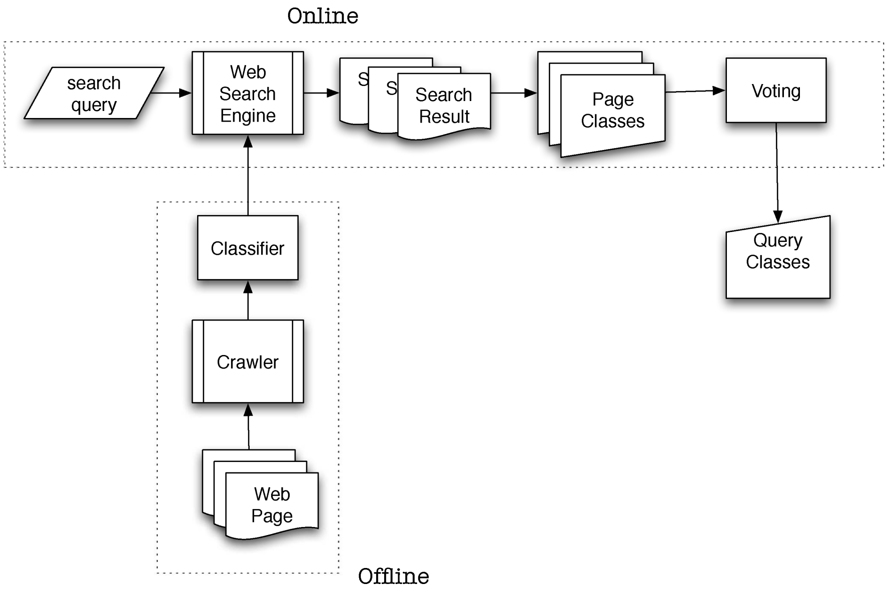


Fig. 1. Query classification architecture.

article, we used a simpler architecture that actually classifies search results on an as-needed basis (see Figure 6 in Section 5). We are currently working on incorporating classification of all indexed pages into the search engine infrastructure, in order to allow fast classification of incoming queries in real time.

—The *Online* component performs actual query classification. Given a query, it sends it to the search engine to obtain a preconfigured number of search results. Then, classifications of individual search results are merged through a voting scheme that computes several classifications for the input query.

4.1 Building the Document Classifier

In this work we used a commercial classification taxonomy of approximately 6000 nodes used in a major US search engine (see Section 5.1). Human editors populated the taxonomy nodes with labeled examples that we used as training instances to learn a document classifier in the offline phase.

Given a taxonomy of this size, the computational efficiency of classification is a major issue. Few machine learning algorithms can efficiently handle so many different classes, each having hundreds of training examples. Suitable candidates include the nearest neighbor and the Naive Bayes classifier [Duda and Hart 1973], as well as prototype formation methods such as Rocchio [1971] or centroid-based [Han and Karypis 2000] classifiers. A recent study [Gabrilovich and Markovitch 2007] showed centroid-based classifiers to be both effective and efficient for large-scale taxonomies; consequently, we used a centroid classifier in this work.

Each centroid is defined as a sum of the TF.IDF values of the terms, normalized by the number of documents in the class $\vec{c_j} = \frac{1}{|C_j|}\sum_{\vec{d}\in C_j}\frac{\vec{d}}{\|\vec{d}\|}$, where $\vec{c_j}$ is the centroid for class C_j, and d iterates over the documents that belong to this class. Classification is based on the cosine of the angle between the document and each of the centroids:

$$C_{max} = \arg\max_{C_j \in C} \frac{\vec{c_j}}{\|\vec{c_j}\|} \cdot \frac{\vec{d_j}}{\|\vec{d_j}\|} = \arg\max_{C_j \in C} \frac{\sum_{i \in |F|} c^i \cdot d^i}{\sqrt{\sum_{i \in |F|} (c^i)^2} \sqrt{\sum_{i \in |F|} (d^i)^2}}, \qquad (1)$$

where F is the set of features. Scores are normalized by the document and class length to make them comparable. The terms c^i and d^i represent the weight of the i-th feature in the class centroid and the document, respectively. These weights are based on the standard "ltc" TF.IDF function (logarithmic term frequency and inverse document frequency, followed by cosine normalization) [Salton and Buckley 1988].

4.2 Query Classification by Search

Having developed a document classifier for the query taxonomy, we now turn to the problem of computing classification(s) for a given query based on the initial search results it yields. Let us assume that there is a set of documents $D = d_1 \dots d_m$ indexed by the search engine. The search engine can then be represented by a function $\vec{f} = \mathit{similarity}(q, d)$ that quantifies the affinity between query q and document d.

Query classification is determined by first evaluating conditional probabilities of all possible classes $P(C_j|q)$, and then selecting the alternative with the highest probability $C_{max} = \mathit{arg}\max_{C_j \in C} P(C_j|q)$. Our goal is to estimate the conditional probability of each possible class using the search results initially returned by the query. We use the following formula that incorporates classifications of individual search results:

$$P(C_j|q) = \sum_{d \in D} P(C_j|q, d) \cdot P(d|q) = \sum_{d \in D} \frac{P(q|C_j, d)}{P(q|d)} \cdot P(C_j|d) \cdot P(d|q).$$

We assume that $P(q|C_j, d) \approx P(q|d)$, that is, the probability of a query given a document can be determined without knowing the class of the query. This is the case for the majority of queries that are unambiguous. Counter examples are queries like 'jaguar' (animal and car brand) or 'apple' (fruit and computer manufacturer), but such ambiguous queries cannot be classified *by definition*, and usually consist of common words. In this work we concentrate on rare queries, that tend to contain rare words, be longer, and match fewer documents; consequently in our setting the assumption mostly holds. Using this assumption, we can write:

$$P(C_j|q) = \sum_{d \in D} P(C_j|d) \cdot P(d|q). \qquad (2)$$

The conditional probability of a classification for a given document $P(C_j|d)$ is estimated using the output of the document classifier (Section 4.1). $P(d|q)$ is harder to compute, and in what follows we consider alternative ways for modeling it.

One simple way to compute $P(d|q)$ is to define $P(d|q) = 1$ if d appears in the top K search results for q, and $P(d|q) = 0$ otherwise; this is reminiscent of the standard relevance feedback approach. Using this definition, we obtain the following simple voting formula $P(C_j|q) = \sum_{d \in D^K(q)} P(C_j|d)$, where $D^K(q)$ is a set of top K search results for q. Here, we first classify each of the top search results, and then use their classes to determine one or several best classes for the query. We call this method *voting* because query classes are effectively voted for by the classes of the search results, based on the strength of the classes in each document.

This method essentially eliminates all information about the document order in the ranked list of search results, hence it is an interesting research question whether taking the ranks into account would ultimately lead to a more accurate estimation of $P(C_j|q)$. This question is addressed in the next sections.

4.3 Classification-Based Relevance Model

In this section we develop a document relevance model based on document and query classes alone (without considering other features of documents and queries, such as individual words). Given a ranked list of documents from the search engine, we then optimize $P(C_j|q)$ so that the output of this restricted relevance model is similar to that produced by the search engine.

Given a document d, and query q, we denote by $R(d, q)$, the relevance of d to q. This value indicates how relevant document d is to query q, and can be used to rank documents for a given query. We now define $R_C(d, q)$, which approximates $R(d, q)$ using a taxonomy of classes:

$$R(d, q) \approx R_C(d, q) = \sum_{C_j \in C} w(C_j) s(C_j, d) s(C_j, q). \tag{3}$$

The right-hand side expresses how we use the classification scheme C to rank documents, where $s(c, d)$ is a scoring function that specifies the likelihood of d belonging to class c, and $s(c, q)$ is a scoring function that specifies the likelihood of q being in class c. The value $w(c)$, is a weighting term for class c, indicating the importance of class c in the relevance formula.

This relevance function is an adaptation of the traditional word-based retrieval approach to class-based features. To observe this, consider what would happen if C_j referred to word occurrence: $s(C_j, d)$ would stand for some function of the word count of C_j in d, $s(C_j, q)$—the word count of C_j in q, and $w(C_j)$—the IDF term weighting for the corresponding word. This way, the method given by (3) becomes the standard TF.IDF retrieval rule.

If we take $s(C_j, d) = P(C_j|d)$, $s(C_j, q) = P(C_j|q)$, and $w(C_j) = 1/P(C_j)$, and assume that q and d are independently generated given a hidden concept C, then:

$$\begin{aligned} R_C(d, q) &= \sum_{C_j \in C} \frac{P(C_j|d) P(C_j|q)}{P(C_j)} \\ &= \sum_{C_j \in C} \frac{P(C_j|d) P(q|C_j)}{P(q)} \end{aligned}$$

$$= \sum_{C_j \in C} \frac{P(q|C_j)P(d|C_j)P(C_j)}{P(d)P(q)}.$$

Assuming the query represents the essence of the class for the purpose of document ranking, we get $P(d|C_j) = P(d|q)$. Upon this substitution, the previous equation transforms into:

$$\begin{aligned} R_C(d,q) &= \sum_{C_j \in C} \frac{P(q|C_j)P(C_j)P(d|q)}{P(d)P(q)} \\ &= \frac{P(d|q)}{P(d)P(q)} \sum_{C_j \in C} P(q|C_j)P(C_j) \\ &= \frac{P(d|q)}{P(d)P(q)} P(q) \\ &= \frac{P(q|d)}{P(q)}. \end{aligned}$$

That is, documents are ranked according to $P(q|d)$. This relevance model has been employed in various statistical language modeling techniques for information retrieval. The intuition can be described as follows. We assume that a user searches for document d by constructing query q: the user first picks a concept C_j, according to the weights $P(C_j|d)$, and then constructs a query q, with probability $P(q|C_j)$, based on the concept C_j. For this query generation process, the documents can be ranked based on the likelihood of the observed query to be generated from each document.

It should be mentioned that in our case, each query and document can have multiple categories. For simplicity, we denote by C_j, a random variable indicating whether q belongs to category C_j. We use $P(C_j|q)$ to denote the probability of q belonging to category C_j. Here the sum $\sum_{C_j \in C} P(C_j|q)$ may not equal to one because the categories are not mutually exclusive, hence a query may be labeled with more than one category. We then consider the following ranking formula:

$$R_C(d,q) = \sum_{C_j \in C} P(C_j|d)P(C_j|q). \tag{4}$$

As before, we assume the estimation of $P(C_j|d)$ is based on an existing text-categorization system (Section 4.1).

In order to obtain estimates of the unknown parameters $P(C_j|q)$, we use Web search results, and assume that the ranking formula (4) gives good ranking for search. That is, we assume that top results ranked by a good Web search engine should also be ranked highly by this formula. Therefore, given query q, and top K result pages $d_1(q), \ldots, d_K(q)$ from a major search engine, we fit parameters $P(C_j|q)$ so that $R_C(d_i(q), q)$ has high scores for $i = 1, \ldots, K$. It should be mentioned that using this method we can only compute relative strength of $P(C_j|q)$, but not the scale, because scale does not affect the ranking. Moreover, it is possible that the estimated parameters may be of the form $g(P(C_j|q))$ for some monotone function $g(\cdot)$ of the actually conditional probability $g(P(C_j|q))$. Although this may change the meaning of the unknown parameters that we estimate, it does not affect the quality of using the formula to rank documents. Nor does it affect query classification with appropriately chosen thresholds. In what follows, we consider two methods to compute the classification information $P(C_j|q)$.

4.4 The Voting Method

First, we show how our classification-based relevance model can be used to derive the simple voting mechanism presented at the end of Section 4.2.

We would like to compute $P(C_j|q)$ so that $R_C(d_i(q), q)$ is high for $i = 1, \ldots, K$ and $R_C(d, q)$ are low for a random document d. Assume that the vector $[P(C_j|d)]_{C_j \in C}$ is random for an average document, then the condition that $\sum_{C_j \in C} P(C_j|q)^2$ is small implies that $R_C(d, q)$ is also small, averaged over d. Thus, a natural method is to maximize $\sum_{i=1}^{K} w_i R_C(d_i(q), q)$ subject to $\sum_{C_j \in C} P(C_j|q)^2$ being small, where w_i are weights associated with each rank i:

$$\max_{[P(\cdot|q)]} \left[\frac{1}{K} \sum_{i=1}^{K} w_i \sum_{C_j \in C} P(C_j|d_i(q)) P(C_j|q) - \lambda \sum_{C_j \in C} P(C_j|q)^2 \right],$$

where we assume $\sum_{i=1}^{K} w_i = 1$, and $\lambda > 0$ is a tuning regularization parameter. The optimal solution is

$$P(C_j|q) = \frac{1}{2\lambda} \sum_{i=1}^{K} w_i P(C_j|d_i(q)). \tag{5}$$

Since both $P(C_j|d_i(q))$ and $P(C_j|q)$ belong to [0, 1], we may just take $\lambda = 0.5$ to align the scale. In our experiment, we will simply take uniform weights $w_i = 1/K$.

4.5 Generalized Voting

Although the voting method in (5) is simple and effective (as we will see in experiments), the choice of the weights w_i, is rather simple. However, it is reasonable to assume that higher ranked documents d_i (with small i), that are more relevant to the query q, should be given a higher weight than lower ranked documents that are less relevant to the query q. That is, in order to optimize (5), we should assign different w_i values for different rank positions i.

Following this argument, a more complex strategy is to let w depend both on d and on q. We may consider the following generalized linear model:

$$\eta(P(C_j|q)) = \sum_{d} w(d, q) g(P(C_j|d)), \tag{6}$$

where $g(x)$ is a certain transformation of x; $\eta(x)$ is a monotone increasing function of x, often referred to as the link function in the statistical literature.

In this general formulation, $w(d, q)$ may depend on factors other than the rank of d in the search engine results for q. For example, it may be a function of $r(d, q)$, where $r(d, q)$ is the relevance score returned by the underlying search engine.

In order to apply (6), it is necessary to know the weight $w(d, q)$, and the transformation $g(P(C_j|d))$. These parameters can be learned using machine learning techniques, if we are provided with a training set of hand-labeled category/query pairs (C_j, q), with binary label y, which indicates whether query q belongs to category C_j. In the experimental section, we apply this idea to a simple model. Specifically, we discretize the quality score $r(d, q)$, of a query/document pair into {high, medium, low}, represented by integers $\{1, 2, 3\}$ respectively. We then learn the three weights w_1, w_2, and w_3, corresponding to these three quality grades. We do not learn the transformation $g(\cdot)$, and simply set it to be the identity. Moreover, we choose the link function $\eta(x) = \ln(x/(1 - x))$, which maps probability in range $(0, 1)$ into the full real line $(-\infty, \infty)$. In statistics, this link function corresponds to logistic regression. Formula (6) then becomes:

$$\ln \frac{P(C_j|q)}{1 - P(C_j|q)} = \sum_d w_{r(d,q)} P(C_j|d). \tag{7}$$

In order to estimate the model parameter w, we need a set of labeled training data $S = \{(q, C_j, y)\}$, where the label $y = 1$ if q belongs to C_j, and $y = -1$ otherwise. We can use the following logistic regression method to estimate $w = [w_1, w_2, w_3]$ in (7):

$$[w_1, w_2, w_3] = \arg\min_w \sum_{(q, C_j, y) \in S} \ln\left(1 + e^{-\sum_d w_{r(d,q)} P(C_j|d) y}\right). \tag{8}$$

4.6 Discriminative Classification

Although the method described in Section 4.5 can be used to estimate parameters in a general weighting formula such as (6), it requires a set of hand-labeled training data, where significant human effort is needed to obtain label y (whether a query belongs to a certain category or not). This section describes another method, which does not require any human-labeled training data. The method is a variation of the voting method in Section 4.4, but with a discriminative classification model.

In this method, we assume, similar to (4), that the search engine relevance function can be approximated by:

$$R_C(d_i(q), q) = \sum_{C_j \in C} P(C_j|q) P(C_j|d_i(q)) = w \cdot x_i(q),$$

where $x_i(q) = [P(C_j|d_i(q))]_{C_j \in C}$ is the feature vector that is known, and $w = [P(C_j|q)]_{C_j \in C}$ is the weight vector that needs to be estimated.

We assume that $R_C(d, q)$ is a good approximation to the search engine relevance function, which means that it approximately preserves the order (or preference) of search engine results. That is, we would like to impose the condition that if $i < j$, then $R_C(d_i(q), q) > R_c(d_i(q), q)$ (document $d_i(q)$, has a higher relevance than document $d_j(q)$). Conceptually this is a very reasonable assumption; however, it may not hold precisely in practice. Although it would be nice to verify to what degree this assumption holds, our data does not provide enough information to do so. Therefore, the only verification in this article is to see whether algorithms derived from this assumption give good results, which we study using experiments. In this sense, its main purpose is rather similar to other statistical assumptions, such as the conditional independence assumption in naive Bayes classification, or the bag-of-word assumption in text modeling—even though it is clear such assumptions do not hold precisely in practice, they are still useful because they lead to interesting algorithms.

We can now treat the problem of estimating $w = [P(C_j|q)]$ as a preference learning problem, which requires the weight vector w to preserve the preference relationship: $i < j$ implies $w \cdot x_i > w \cdot x_j$. A more specific and simpler method, which we employ in this article, is to treat the problem of estimating $w = [P(C_j|q)]$ as a classification problem. For each q, we label $d_i(q)$ for $i = 1, \ldots, K$ as positive data, and the remaining documents as negative data. That is, we assign label $y_i(q) = 1$ for $d_i(q)$ when $i \leq K$, and label $y_i(q) = -1$ for $d_i(q)$ when $i > K$. This is equivalent to enforcing the preference relationship $w \cdot x_i > w \cdot x_j$ if $i \leq K$ and $j > K$. Note that although we have labels $y_i(q)$, they are automatically obtained from search engine results without any human effort. This is an advantage over the method in Section 4.5.

In this classification formulation, the values $P(C_j|d)$ $(C_j \in C)$ are the features for the linear classifier, and $w = [P(C_j|q)]$ is the weight vector, which can be estimated using any linear classification method. In this article, we consider estimating w using logistic regression [Santner and Duffy 1989] as follows:

$$P(\cdot|q) = \arg\min_{w} \sum_{i} \ln\left(1 + e^{-w \cdot x_i(q) y_i(q)}\right).$$

That is, the desired probability estimate $P(C_j|q)$ for each $C_j \in C$ is a coefficient of this logistic regression solution.

5. EVALUATION

In this section, we evaluate our methodology that uses Web search results for improving query classification.

5.1 Taxonomy

Our choice of taxonomy was guided by a Web advertising application. Since we want the classes to be useful for matching ads to queries, the taxonomy needs to be elaborate enough to facilitate ample classification specificity. For example, classifying all medical queries into one node will likely result in poor ad matching, as both "sore foot" and "flu" queries will end up in the same node. The ads appropriate for these two queries are, however, very different. To avoid such situations, the taxonomy needs to provide sufficient discrimination between common commercial topics. Therefore in this article, we employ an elaborate taxonomy of approximately 6000 nodes, arranged in a hierarchy with median depth 5 and maximum depth 9.

The taxonomy has been populated with labeled training examples that were bid phrases of actual ads. Ideally, it is preferable to have labeled training documents from the same distribution from which documents to be classified are drawn. Since our method classifies Web search results, labeled training

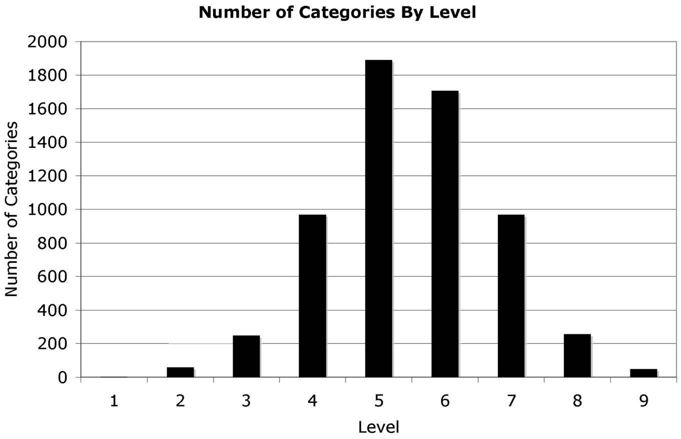


Fig. 2. Taxonomy statistics: number of categories per level.

examples should ideally also be Web pages. It is, however, prohibitively expensive to manually label a large enough set of Web pages at the resolution we need (to populate a taxonomy of 6,000 nodes). It is substantially cheaper to label short bid phrases rather than long documents. Using labeled bid phrases of ads is also particularly suitable for our application, since our research is motivated by the need to match queries to more relevant ads.

The taxonomy has been populated by human editors using keyword suggestion tools similar to the ones used by ad networks to suggest keywords to advertisers. After initial seeding with a few queries, using the provided tools, a human editor can add several hundred queries to a given node, which were used as a training set. A small fraction of queries have been assigned to more than one category.[2] Nevertheless, it has been a significant effort to develop a taxonomy of a magnitude of several person-years. A similar-in-spirit process for building enterprise taxonomies via queries has been presented in Gates et al. [2005]. However, the details and tools are completely different. Figures 2 and 3 show pertinent statistics about the structure of the taxonomy, and Figures 4 and 5 show statistics about the labeled examples used to train the classifier described in Section 4.1.

5.2 Data Sets

We used two representative sets of 1000 queries. Both sets contain queries that cannot be directly matched to advertisements, that is, none of the queries contains a bid phrase (this means we eliminated practically all popular queries).

[2] Some queries were assigned to more than one category because they had several equally important facets. For example, a query about antivirus software for Linux could be simultaneously assigned to categories *"Computing / Computer Security / Malicious Software Prevention and Elimination / Virus Utilities / Anti Virus Utilities - Linux"* and *"Computing / Computer Software / Software Utilities / Security Software / Firewalls / Firewalls - Linux"*. Here, the former classification emphasizes the security application, and the latter—the fact that the application is implemented in software rather than in hardware.

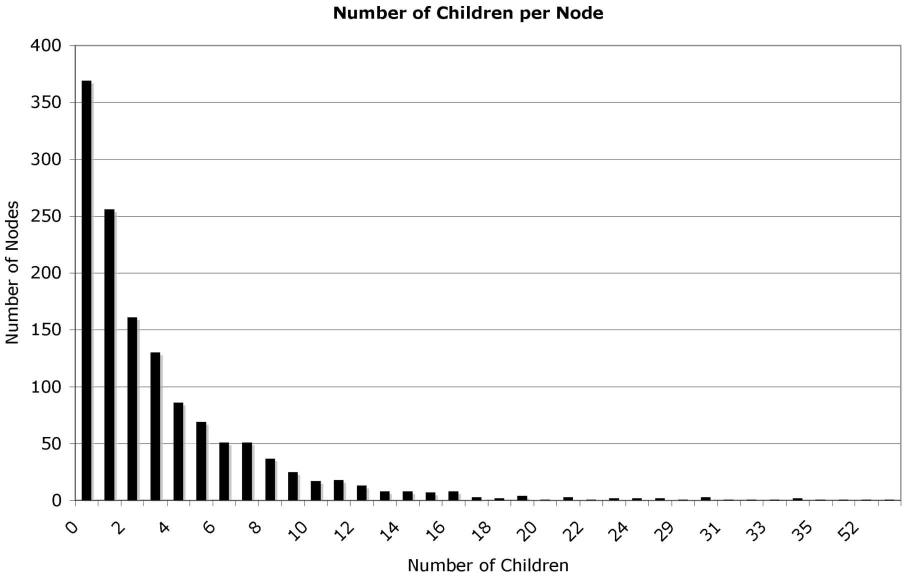


Fig. 3. Taxonomy statistics: fanout of non-leaf nodes.

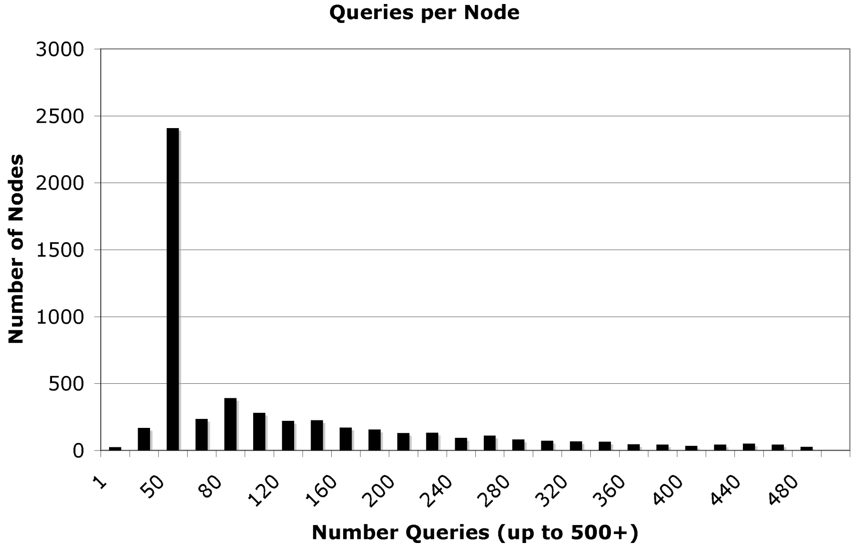


Fig. 4. Taxonomy statistics: number of training examples (queries) per node.

The first set of queries can be matched to at least one ad using *broad* match as previously described. Queries in the second set cannot be matched even by broad match, therefore the search engine used in our study does not currently display any advertising for them. In a sense, these are even rarer queries and further away from common queries. As a measure of query rarity, we estimated their frequency in a month's worth of query logs for a major US search engine; the median frequency was 1 for queries in Set 1 and 0 for queries in Set 2.

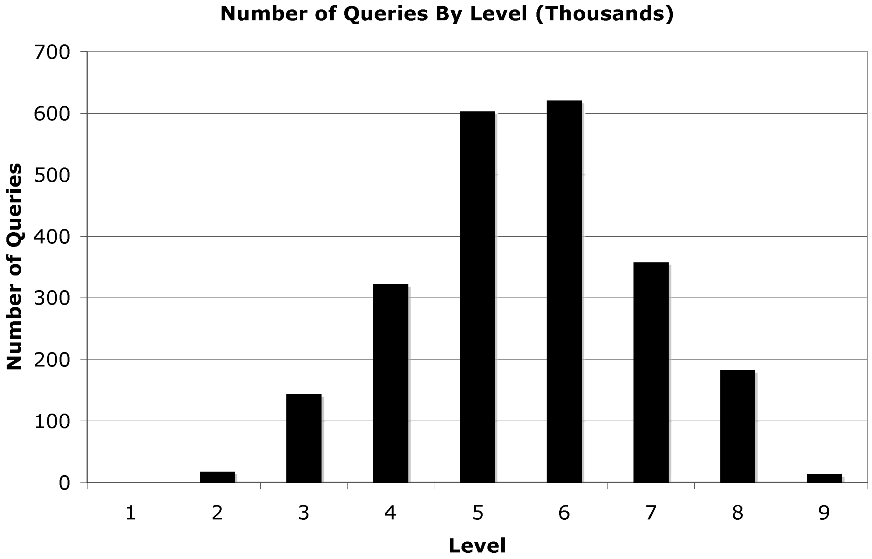


Fig. 5. Taxonomy statistics: number of training examples (queries) per level.

The queries in the two sets differ in their classification difficulty. In fact, queries in Set 2 are difficult to interpret even for human evaluators. Queries in Set 1 have on average 3.50 words, with the longest one having 11 words; queries in Set 2 have on average 4.39 words, with the longest query having 81 words. Recent studies estimate the average length of Web queries to be just under 3 words,[3] which is lower than in our test sets. As another measure of query difficulty, we measured the fraction of queries that contain quotation marks, as the latter assist query interpretation by meaningfully grouping the words. Only 8% of queries in Set 1 and 14% in Set 2 contained quotation marks.

The queries in the two test sets, Set 1 and Set 2, are different from the set of labeled examples used to train the classifier, as described in Section 4.1, for two reasons. First, in order to have a test set that is disjoint from the training set. Second, in this study we are primarily interested in classifying rare queries, while the taxonomy has been populated with sufficiently frequent queries.

5.3 Methodology and Evaluation Metrics

The two sets of queries were classified into the target taxonomy using the techniques presented in Section 4. Based on the confidence values assigned, the top 3 classes for each query were presented to human evaluators. These evaluators were trained editorial staff who possessed knowledge about the taxonomy. The editors considered every query-class pair, and rated them on the scale 1 to 4, with 1 meaning the classification is highly relevant and 4 meaning it is irrelevant for the query. About 2.4% of queries in Set 1 and 5.4% of queries in Set 2 were judged to be unclassifiable (e.g., random strings of characters), and

[3] http://www.rankstat.com/html/en/seo-news1-most-people-use-2-word-phrases-in-search-engines.html

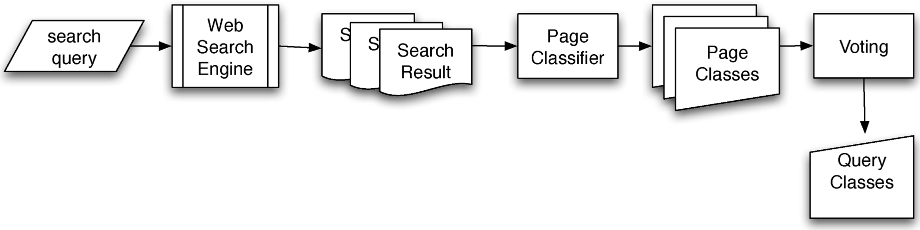


Fig. 6. Query classification architecture (prototype).

were consequently excluded from evaluation. To compute evaluation metrics, we treated classifications with ratings 1 and 2 to be correct, and those with ratings 3 and 4 to be incorrect. For each query, the human judges assessed the quality of up to three top-scoring classes assigned according to Equation (1), which were produced by four different instantiations of the classifier (the four instantiations differ by the parameters discussed in the next section). This process required collecting up to 2,000 × 3 × 4 = 24,000 judgments. After excluding the unclassifiable queries as well as duplicate classifications produced by the different algorithms, we obtained 21,260 judgments for query-class pairs. Three human judges participated in the experiment, however, each query-class pair was judged by a single person owing to cost considerations. It should be noted that the judges were highly trained, and their inter-editor agreement measured on similar tasks in the past was over 70%.

We used standard evaluation metrics: precision, recall, and micro-F1. Precision is the proportion of actual positive class members returned by the classifier among all the predicted positive class members returned. Recall is the proportion of predicted positive members among all actual positive class members in the result set.[4] Since we aim to maximize both precision and recall, a combined measure of the two is useful to evaluate the overall performance of the classifier. The F1 measure is the harmonic mean of precision P and recall R, defined as $F1 = 2 * P * R/(P + R)$. To compute micro-averaged F1, both precision and recall are computed for the entire test collection (rather than individually for different categories, as would be the case for macro-averaging). In what follows, we plot precision-recall graphs for all the experiments. For comparison with other published studies, we also report precision and F1 values corresponding to complete recall ($R = 1$).

As explained in Section 4, query classification would be most efficient if all the pages in the search index have been preclassified. To achieve this aim, however, it would be necessary to significantly extend the crawling infrastructure of the search engine to perform Web page classification at crawling time. Since this mechanism was not available at the time of our study, we instead opted to build our prototype, which classifies search results for each query on the fly. Figure 6 outlines the prototype infrastructure we used in our work.

[4] When computing recall, we treat the set of all correct classes among these three judged ones for each query as corresponding to full recall. Therefore, the recall in the figures can actually reach 1.0 (= 100% recall).

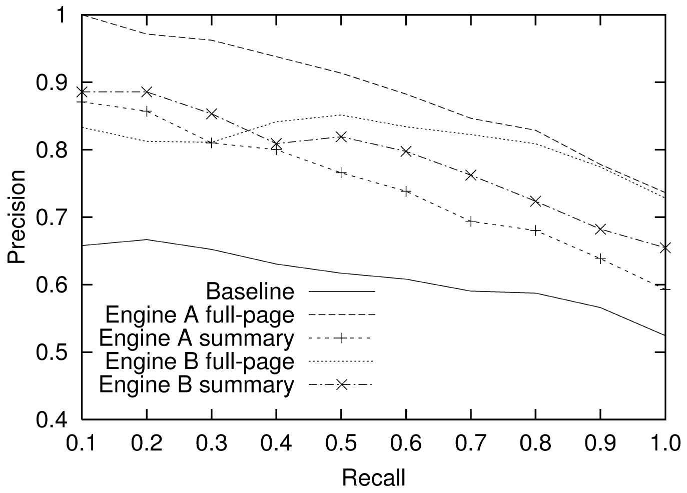


Fig. 7. The effect of external knowledge (merged Set 1 and Set 2).

5.4 Results

We compared our method to a baseline query classifier that does not use any external knowledge. Our baseline classifier expanded queries using standard query expansion techniques, grouped their terms using a phrase recognizer, boosted certain phrases in the query based on their statistical properties, and performed classification using the nearest-neighbor approach. The same training data was used for the baseline as for the proposed method. This baseline classifier is actually a production version of the query classifier running in a major US search engine.

In our experiments, we varied values of pertinent parameters that characterize the exact way of using search results. In what follows, we start with the general assessment of the effect of using Web search results supplied by two different major US search engines (henceforth denoted as A and B). In what follows, Figure 7 shows the results using both search engines, while the other figures show the results using only engine B. We then proceed to exploring more refined techniques, such as using only search summaries versus actually crawling the returned URLs. We also experimented with using different numbers of search results per query, as well as with varying the number of classifications considered for each search result.

5.4.1 *The Effect of External Knowledge.* Queries by themselves are very short and difficult to classify. We use top search engine results in the form of either summaries or full text pages of the top hits for collecting background knowledge for queries. We employed two major US search engines, and used their results in two ways, either only summaries or the full text of crawled result pages. Figure 7 and Table I show that such extra knowledge considerably improves classification accuracy.

Table I. The Effect of Using External Knowledge (Precision and F1 are Computed at Full Recall)

Engine	Context	Precision	F1	Precision	F1
		Set 1	Set 1	Set 2	Set 2
A	full-page	0.72	0.84	0.509	0.721
B	full-page	0.706	0.827	0.497	0.665
A	summary	0.586	0.744	0.396	0.572
B	summary	0.645	0.788	0.467	0.638
Baseline		0.534	0.696	0.365	0.536

Table II. Aggregation: Bundling vs. Individual Voting (Precision and F1 are Computed at Full Recall)

Setting	Context	Precision	F1	Precision	F1
		Set 1	Set 1	Set 2	Set 2
Bundled	full-page	0.678	0.811	0.492	0.662
Bundled	summary	0.665	0.801	0.478	0.648
Voting	full-page	**0.706**	**0.828**	**0.497**	**0.665**
Voting	summary	0.645	0.788	0.467	0.638
baseline		0.534	0.696	0.365	0.536

5.4.2 *Aggregation Techniques.* There are two major ways to use search results as additional knowledge. First, individual results can be classified separately, with subsequent voting among individual classifications. Alternatively, individual search results can be bundled together as one metadocument and classified as such, using the document classifier. Table II and Figures 8 and 9 present the results of these two approaches for classifying search results individually versus concatenating them together into a single metadocument. When full-text pages are used, the technique using individual classifications of search results evidently outperforms the bundling approach by a wide margin. However, in the case of summaries, bundling together is found to be consistently better than individual classification. This is because summaries by themselves are too short to be classified correctly individually, but when bundled together they are much more stable. In query Set 2, the difference between the techniques is even more pronounced. As can be seen in Figure 9, for full pages, individual voting outperforms bundled aggregation by as much as 25%, while in the case of summaries, bundling is better by about 10%.

5.4.3 *Full Page Text versus Summary.* To summarize the two preceding sections, background knowledge for each query is obtained by using either the full-page text or only the summaries of the top search results. Full page text was found to be more useful in conjunction with voted classification, while summaries were found to be useful when bundled together. The best results overall were obtained with full-page results classified individually, with subsequent voting used to determine the final query classification. This observation differs from findings by Shen et al. [2006b], who found summaries to be more useful. We attribute this distinction to the fact that the queries we used in this study are tail ones, which are rare and difficult to classify.

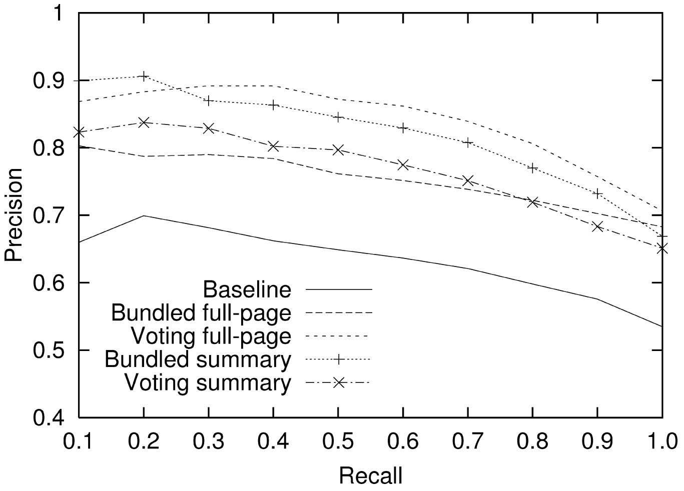


Fig. 8. Voting vs. bundling (Set 1).

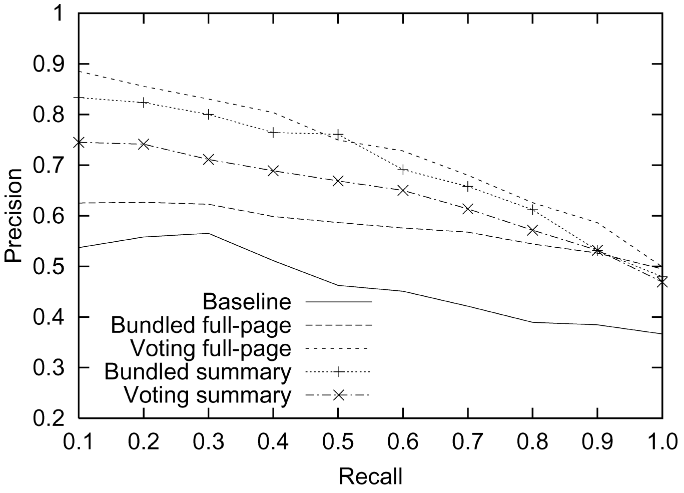


Fig. 9. Voting vs. bundling (Set 2).

5.4.4 *Varying the Number of Classes per Search Result.* We also varied the number of classifications per search result: each result was permitted to have either one, three, or five classes. Figure 10 shows the corresponding precision-recall graphs for both full-page and summary-only settings. As can be readily seen, all three variants produce very similar results. However, the precision-recall curve for the one-class experiment has higher fluctuations. Using three classes per search result yields a more stable curve, while with five classes per result the precision-recall curve is very smooth. Thus, as we increase the number of classes per result, we observe higher stability in query classification. This happens because as we increase the number of classes, the influence of each individual vote towards a particular class is reduced and smoothed out over the aggregation.

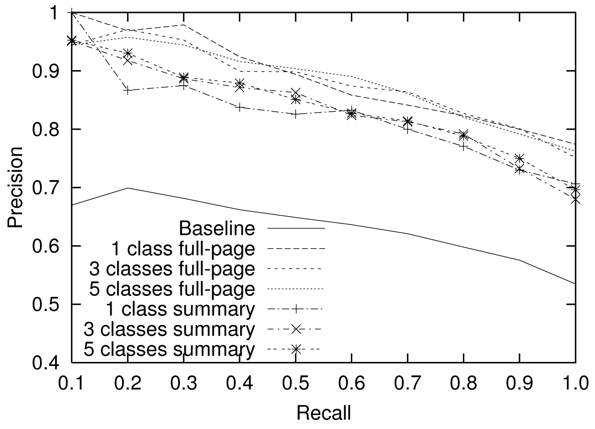


Fig. 10. Varying the number of classes per page (Set 1).

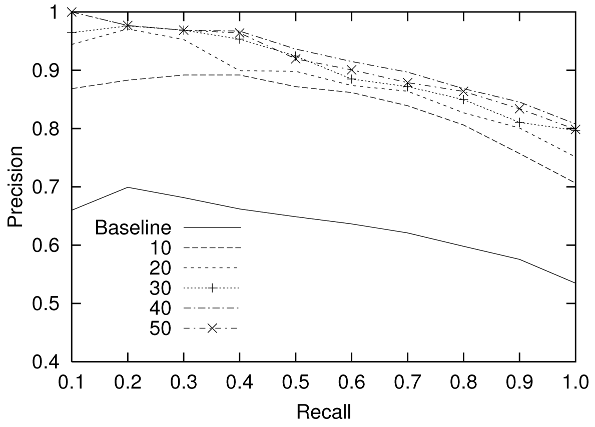


Fig. 11. Varying the number of results per query (Set 2).

5.4.5 *Varying the Number of Search Results Obtained.* We also experimented with different numbers of search results per query. Figure 11 and Table III present the results of this experiment. In line with our intuition, we observed that classification accuracy steadily rises as we increase the number of search results used, from 10 to 40, with a slight drop as we continue to use even more results (50). This is because using too few search results provides too little external knowledge, while using too many results introduces extra noise.

Using paired t-test, we assessed the statistical significance of the improvements due to our methodology versus the baseline. We found the results to be highly significant ($p < 0.0005$), thus confirming the value of external knowledge for query classification.

5.5 Voting versus Alternative Methods

As explained in Section 4.2, one may use several methods to classify queries from search engine results based on our relevance model. As we have seen, the

Table III. Varying the Number of Search Results

Number of Results	Precision	F1
baseline	0.534	0.696
10	0.706	0.827
20	0.751	0.857
30	0.796	0.886
40	**0.807**	**0.893**
50	0.798	0.887

voting method works quite well. In this section, we compare the performance of voting among the top-ten search results to the following two methods:

— GV: Generalized voting method described in Section 4.5. We discretize the quality score $r(d, q)$ of a query/document pair (returned by a search engine) into {high, medium, low}. We then learn the three corresponding weights on a set of training queries using (8), and test the performance on holdout queries.

— DC: Discriminative classification learning of query-classification based on logistic regression, described in Section 4.6.

GV requires a training/testing split. Neither voting nor DC requires such a split; however, for consistency, we randomly draw 50-50 training/testing splits for ten times, and report the mean performance ± standard deviation on the test-split for all methods. For this experiment, instead of precision and recall, we use DCG-k, popular in search engine evaluation. The DCG (discounted cumulative gain) metric, described in Jarvelin and Kekalainen [2000], is a ranking measure where the system is asked to rank a set of candidates (in our case, judged categories for each query), and computes for each query q:

$$DCG_k(q) = \sum_{i=1}^{k} g(C_i(q))/\log_2(i+1),$$

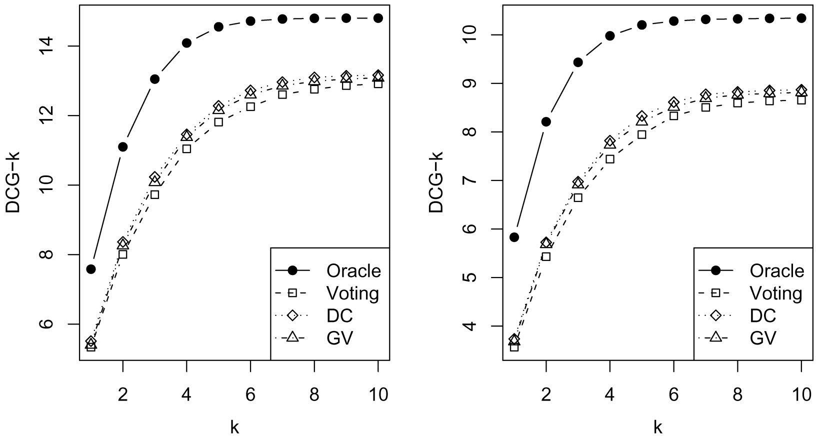
where $C_i(q)$ is the i-th category for query q ranked by the system, and $g(C_i)$ is the grade of C_i: we assign grades of $10, 5, 1, 0$ to the 4-point judgment scale described earlier to compute DCG. The decaying choice of $\log_2(i+1)$ is conventional, which does not have particular importance. The overall DCG of a system is the DCG averaged over queries. We use this metric instead of precision/recall in this experiment because it can directly handle multigrade outputs. Therefore as a single metric, it is convenient for comparing different methods. Note that the precision/recall curves used in the earlier sections yield some additional insights not immediately apparent from the DCG numbers.

Results from our experiments are given in Table IV, where DCG-1 and DCG-5 are reported with mean ± standard deviation. Figure 12 shows DCG-k for $k = 1, \ldots, 10$, where we only report the mean values over the ten random splits. The oracle method is the best ranking of categories for each query after seeing human judgments. It cannot be achieved by any realistic algorithm, but is included here as an absolute upper bound on DCG performance. The simple voting method performs very well in our experiments. The more complicated

Table IV. Voting and Alternative Methods

Set 1		
Method	DCG-1	DCG-5
Oracle	7.58 ± 0.12	14.56 ± 0.19
Voting	5.33 ± 0.16	11.82 ± 0.21
DC	5.51 ± 0.10	12.28 ± 0.19
GV	5.40 ± 0.13	12.16 ± 0.19

Set 2		
Method	DCG-1	DCG-5
Oracle	5.83 ± 0.10	10.21 ± 0.08
Voting	3.56 ± 0.14	7.94 ± 0.20
DC	3.74 ± 0.10	8.33 ± 0.12
GV	3.68 ± 0.06	8.21 ± 0.13



Set 1

Set 2

Fig. 12. DCG-k for $k = 1, \ldots, 10$.

methods may lead to moderate performance gain (especially DC, which uses discriminative training in Section 4.6). However, both methods are computationally more costly, and the potential gain is minor enough to be neglected. This means that as a simple method, voting is quite effective.

We can observe that GV, which uses a quality score returned by a search engine to adjust importance weights of returned pages for a query, does not yield appreciable improvement. This implies that putting equal weights (voting) performs similarly as putting higher weights to higher quality documents and lower weights to lower quality documents (GV), at least for the top search results. It may be possible to improve this method by including other page-features that can differentiate top-ranked search results. However, the effectiveness will require further investigation, which we did not test. We may also observe that the performance on Set 2 is lower than that on Set 1, which means queries in Set 2 are harder than those in Set 1.

5.6 Failure Analysis

We scrutinized the cases when external knowledge did not improve query classification, and identified three main causes for such lack of improvement. (1) Queries containing random strings, such as telephone numbers—these queries do not yield coherent search results, and so the latter cannot help classification (around 5% of queries were of this kind). (2) Queries that yield no search results at all; there were 8% such queries in Set 1 and 15% in Set 2. (3) Queries corresponding to recent events, for which the search engine did not yet have ample coverage (around 5% of queries). One notable example of such queries are entire names of news articles—if the exact article has not yet been indexed by the search engine, search results are likely to be of little use.

6. CONCLUSIONS

Query classification is an important information retrieval task. Accurate classification of search queries can potentially be useful in a number of higher-level tasks such as Web search and ad matching. Since search queries are usually short, by themselves they usually carry insufficient information for adequate classification accuracy. To address this problem, we proposed a methodology for using search results as a source of external knowledge. To this end, we send the query to a search engine, and assume that a plurality of the highest-ranking search results are relevant to the query. Classifying these results then allows us to classify the original query with substantially higher accuracy.

The results of our empirical evaluation definitively confirmed that using the Web as a repository of world knowledge contributes valuable information about the query, and aids in its correct classification. Notably, our method exhibits significantly higher accuracy than methods described in prior studies.[5] Compared to earlier studies, our approach does not require any auxiliary taxonomy, and we produce a query classifier directly for the target taxonomy. Furthermore, the taxonomy used in this study is approximately two orders of magnitude larger than that used in prior works.

We also experimented with different values of parameters that characterize our method. When using search results, one can either use only summaries of the results provided by the search engine, or actually crawl the results pages for even deeper knowledge. Overall, query classification performance was best when using the full crawled pages (Table I). These results are consistent with prior studies [Gabrilovich and Markovitch 2007], which found that using full crawled pages is superior for document classification than using only brief summaries. Our findings, however, are different from those reported by Shen et al. [2006a], who found summaries to yield better results. We attribute our observations to using a more elaborate voting scheme among the classifications of individual search results, as well as to using a more difficult set of rare queries.

We also found that the best results were obtained by using full crawled pages and performing voting among their individual classifications. For a classifier

[5] Since the field of query classification does not yet have established and agreed-upon benchmarks, direct comparison of results is admittedly tricky.

that is external to the search engine, retrieving full pages may be prohibitively costly, in which case one might prefer to use summaries to gain computational efficiency. On the other hand, for the owners of a search engine, full page classification is much more efficient, since it is easy to preprocess all indexed pages by classifying them once onto the fixed taxonomy. Then, page classifications can be obtained as part of the metadata associated with each search result, and query classification can be nearly instantaneous.

When using summaries it appears that better results are obtained by first concatenating individual summaries into a metadocument, and then using its classification as a whole. We believe the reason for this observation is that summaries are short and inherently noisier, hence their aggregation helps to correctly identify the main theme. Consistent with our intuition, using too few search results yields useful but insufficient knowledge, and using too many search results leads to inclusion of marginally relevant Web pages. The best results were obtained when using 40 top search hits.

In this work, we first classify search results, and then use their classifications directly to classify the original query. Alternatively, one can use the classifications of search results as *features* in order to learn a second-level classifier. In Section 5.5, we reported some preliminary experiments in this direction, and found that learning such a secondary classifier did not yield considerable advantages. We plan to further investigate this direction in our future work.

It is also essential to note that implementing our methodology incurs little overhead. If the search engine classifies crawled pages during indexing, then at query time we only need to fetch these classifications and do the voting.

Our methodology for using search results can be particularly beneficial for rare queries, for which little per-query learning can be done. In the present study we proved that such scarceness of information could be addressed by leveraging the knowledge found on the Web. In our recent work [Broder et al. 2009, 2008] we used the methodology developed in this article to match Web search queries with more relevant ads. To date, such matching has mostly been performed using the bag of words, and we showed that using classification-based features allows us to perform more fine-grained matching and serve more relevant ads.

In our further research we plan to make use of session information in order to leverage knowledge about previous queries to better classify subsequent ones. We are also studying the effect of query classification in new domains such as folksonomies, where the context for query classification consists of tags assigned by the users to cataloged objects.

ACKNOWLEDGMENTS

We thank Marti Hearst for thoughtful comments that helped improve this article.

REFERENCES

BEITZEL, S., JENSEN, E., CHOWDHURY, A., GROSSMAN, D., AND FRIEDER, O. 2004. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, 321–328.

Beitzel, S., Jensen, E., Frieder, O., Grossman, D., Lewis, D., Chowdhury, A., and Kolcz, A. 2005a. Automatic web query classification using labeled and unlabeled training data. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*. ACM Press, 581–582.

Beitzel, S., Jensen, E., Frieder, O., Lewis, D., Chowdhury, A., and Kolcz, A. 2005b. Improving automatic query classification via semi-supervised learning. In *Proceedings of the 5th IEEE International Conference on Data Mining*. IEEE Computer Society.

Beitzel, S. M., Jensen, E. C., Lewis, D. D., Chowdhury, A., and Frieder, O. 2007. Automatic classification of web queries using very large unlabeled query logs. *ACM Trans. Inform. Syst. 25*, 1–29.

Broder, A., Ciccolo, P., Fontoura, M., Gabrilovich, E., Josifovski, V., and Riedel, L. 2008. Search advertising using Web relevance feedback. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*.

Broder, A., Ciccolo, P., Gabrilovich, E., Josifovski, V., Metzler, D., Riedel, L., and Yuan, J. 2009. Online expansion of rare queries for sponsored search. In *Proceedings of the 18th International World Wide Web Conference*.

Broder, A., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V., and Zhang, T. 2007. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th ACM International Conference on Research and Development in Information Retrieval*. ACM Press.

Duda, R. and Hart, P. 1973. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, NY.

Efthimiadis, E. and Biron, P. 1994. UCLA-Okapi at TREC-2: Query expansion experiments. In *Proceedings of the Text REtrieval Conference (TREC-2)*. National Institute of Standards and Technology (NIST).

Gabrilovich, E. and Markovitch, S. 2007. Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization. *J. Mach. Learn. Res. 8*, 2297–2345.

Gates, S. C., Teiken, W., and Cheng, K.-S. F. 2005. Taxonomies by the numbers: building high-performance taxonomies. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM Press, 568–577.

Gravano, L., Hatzivassiloglou, V., and Lichtenstein, R. 2003. Categorizing web queries according to geographical locality. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM)*. ACM Press, 325–333.

Han, E. and Karypis, G. 2000. Centroid-based document classification: Analysis and experimental results. In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*. Springer-Verlag.

Jarvelin, K. and Kekalainen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the ACM Conference and Research and Development in Information Retrieval (SIGIR)*. ACM Press.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*. 137–142.

Kardkovacs, Z., Tikk, D., and Bansaghi, Z. 2005. The ferrety algorithm for the KDD Cup 2005 problem. In *SIGKDD Explorations*. Vol. 7. ACM, 111–116.

Kowalczyk, P., Zukerman, I., and Niemann, M. 2004. Analyzing the effect of query class on document retrieval performance. In *Proceedings of the Australian Conference on Artificial Intelligence*. Springer, 550–561.

Li, Y., Zheng, Z., and Dai, H. 2005. KDD CUP-2005 report: Facing a great challenge. In *SIGKDD Explorations*. Vol. 7. ACM, 91–99.

Lu, Y., Peng, F., Li, X., and Ahmed, N. 2006. Coupling feature selection and machine learning methods for navigational query identification. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM Press, 682–689.

Manning, C. D., Raghavan, P., and Schuetze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

McCallum, A. and Nigam, K. 1998. A comparison of event models for naive Bayes text classification. In *AAAI/ICML Workshop on Learning for Text Categorization*. 41–48.

MITRA, M., SINGHAL, A., AND BUCKLEY, C. 1998. Improving automatic query expansion. In *Proceedings of the 21st ACM International Conference on Research and Development in Information Retrieval (SIGIR)*. ACM Press, 206–214.

MORAN, M. AND HUNT, B. 2005. *Search Engine Marketing, Inc.: Driving Search Traffic to Your Company's Web Site*. Prentice Hall, Upper Saddle River, NJ.

ROBERTSON, S., WALKER, S., JONES, S., HANCOCK-BEAULIEU, M., AND GATFORD, M. 1995. Okapi at TREC-3. In *Proceedings of the Text REtrieval Conference (TREC-3)*. NIST, Gaithersburg, MD.

ROCCHIO, J. 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, 313–323.

SAHAMI, M., MITTAL, V., BALUJA, S., AND ROWLEY, H. 2004. The happy searcher: Challenges in web information retrieval. In *Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence*. Springer-Verlag.

SALTON, G. AND BUCKLEY, C. 1988. Term weighting approaches in automatic text retrieval. *Inform. Proc. Manag. 24,* 5, 513–523.

SALTON, G. AND BUCKLEY, C. 1990. Improving retrieval performance by relevance feedback. *J. Am. Soc. Inform. Sci. 41,* 4, 288–297.

SANTNER, T. AND DUFFY, D. 1989. *The Statistical Analysis of Discrete Data*. Springer-Verlag.

SEBASTIANI, F. 2002. Machine learning in automated text categorization. *ACM Comput. Surv. 34,* 1, 1–47.

SHEN, D., PAN, R., SUN, J., PAN, J., WU, K., YIN, J., AND YANG, Q. 2005. Q2C@UST: Our winning solution to query classification in KDDCUP 2005. In *SIGKDD Explorations*. Vol. 7. ACM, Chicago, IL, USA, 100–110.

SHEN, D., PAN, R., SUN, J., PAN, J., WU, K., YIN, J., AND YANG, Q. 2006a. Query enrichment for web-query classification. *ACM Trans. Info. Syst. 24*, 320–352.

SHEN, D., SUN, J., YANG, Q., AND CHEN, Z. 2006b. Building bridges for web query classification. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*. ACM Press, 131–138.

VOGEL, D., BICKEL, S., HAIDER, P., SCHIMPFKY, R., SIEMEN, P., BRIDGES, S., AND SCHEFFER, T. 2005. Classifying search engine queries using the web as background knowledge. In *SIGKDD Explorations*. Vol. 7. ACM, 117–122.

VOORHEES, E. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*. Springer-Verlag, 61–69.

XU, J. AND CROFT, W. B. 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inform. Sys. 18,* 1, 79–112.

YANG, Y. 1999. An evaluation of statistical approaches to text categorization. *Inform. Retriev. J. 1*, 69–90.

ZHANG, T. AND OLES, F. J. 2001. Text categorization based on regularized linear classification methods. *Inform. Retriev. 4*, 5–31.

Received February 2008; revised August 2008, December 2008; accepted February 2009