Query Enrichment for Web-Query Classification

DOU SHEN and RONG PAN
Hong Kong University of Science and Technology
JIAN-TAO SUN
Microsoft Research Asia
and
JEFFREY JUNFENG PAN, KANGHENG WU, JIE YIN, and QIANG YANG
Hong Kong University of Science and Technology

Web-search queries are typically short and ambiguous. To classify these queries into certain target categories is a difficult but important problem. In this article, we present a new technique called query enrichment, which takes a short query and maps it to intermediate objects. Based on the collected intermediate objects, the query is then mapped to target categories. To build the necessary mapping functions, we use an ensemble of search engines to produce an enrichment of the queries. Our technique was applied to the ACM Knowledge Discovery and Data Mining competition (ACM KDDCUP) in 2005, where we won the championship on all three evaluation metrics (precision, F1 measure, which combines precision and recall, and creativity, which is judged by the organizers) among a total of 33 teams worldwide. In this article, we show that, despite the difficulty of an abundance of ambiguous queries and lack of training data, our query-enrichment technique can solve the problem satisfactorily through a two-phase classification framework. We present a detailed description of our algorithm and experimental evaluation. Our best result for F1 and precision is 42.4% and 44.4%, respectively, which is 9.6% and 24.3% higher than those from the runner-ups, respectively.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Query classification, query enrichment, synonym-based classifier, ensemble learning, KDDCUP2005

This work was supported by NEC China Lab under Grant NECLC05/06.EG01, and Hong Kong RGC Grant HKUST 6187/04E.
Authors' addresses: D. Shen, R. Pan, J. J. Pan, K. Wu, J. Yin, and Q. Yang, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China; email: {dshen,panrong,panjf,khwu,yinjie,qyang}@cse.ust.hk; J.-T. Sun, Microsoft Research Asia, 49 Zhichun Road, Beijing, China; email: jtsun@microsoft.com.

1. INTRODUCTION

Historically, search engine technologies and automatic text-classification techniques have progressed hand-in-hand. Ever since the early articles by the pioneers [Jones 1971; Lewis and Gale 1994], people have recognized the possibility of conducting both web search through classification, and vice versa [Page et al. 1998; Beeferman and Berger 2000; Chekuri et al. 1997; Chen and Dumais 2000; Kang and Kim 2003]. The 2005 ACM Knowledge Discovery and Data Mining competition (KDDCUP2005, for short) made this connection even stronger. This competition series (publicly available at http://www.acm.org/sigs/sigkdd/kddcup/), which has a long history dating back to 1997, is open to researchers and practitioners worldwide. Being one of the most prominent data mining competitions, the datasets used therein are often employed later as benchmarks. The task in KDDCUP2005 is to accurately automatically classify a subset of query-log data from one month in 2005 out of the MSN search engine (http://search.msn.com), one of the major search engines, into a set of given target categories. The log data contains 800,000 queries and the target categories consist of 67 predetermined classes provided by the organizers. The dataset is available as a public domain benchmark for query classification (http://www.acm.org/sigs/sigkdd/kdd2005/kddcup.html). Several example queries are shown in Table I. An illustration of the hierarchical structure for the target categories is shown in Figure 1 (see Appendix A for details).

The KDDCUP2005 task highlights the importance and difficulties of query classification, which is a way to understand queries [Li et al. 2005; Shen et al. 2005; Kardkovács et al. 2005; Vogel et al. 2005; Shen et al. 2006]. Query classification can be used to support several important tasks in information retrieval and Web search. In information retrieval, a potential area is to construct user models so as to cater to both individual and group user preferences. The classification of user queries is a component of both constructing and utilizing user models. In Web search, an important application is to organize the large number of Web pages in the search result after the user issues a query, according to the potential categories of the results. Query classification can be used to effectively organize these results. Furthermore, in Web search, many search engine companies are interested in providing commercial services in response to user queries, including targeted advertisement, product reviews, and valued-added services such as banking and transportation, according to the categories. In these applications of Web search, query classification is very important. Instead of classifying queries directly, some previous work focuses on classifying search results as an alternative way to understand queries [Chen and Dumais 2000].

However, there are several major difficulties which hinder the progress of query classification. First, many queries are short and query terms can be noisy. As an example, in the KDDCUP2005 dataset the 800,000 queries vary a lot. They can be as simple as a single number, such as “1939,” or as complicated as a piece of programming code involveing more than 50 words. Figure 2 shows statistical information about the number of words contained in each query and their frequencies in the 800,000 queries. From this figure we can see

Table I. Examples of Queries

1967 shelby mustang
actress hildegarde
Aldactone
alfred Hitchcock
amazon rainforest
section8rentalhouses.com
Sakpsabancnnhayat
auto price
a & r management" property management Maryland
netconfig.exe

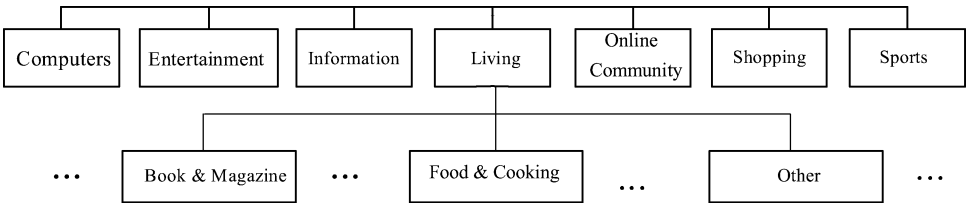


Fig. 1. Illustration of the hierarchy structure of 67 target categories.

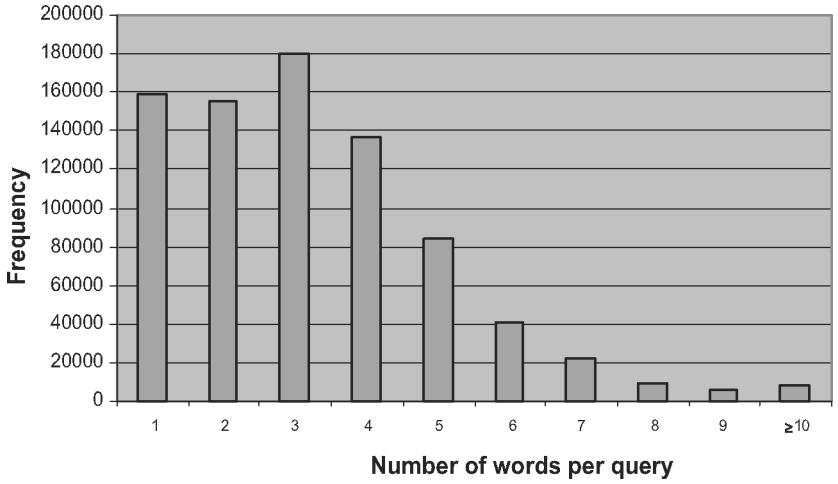


Fig. 2. Frequency of queries with different lengths.

that queries containing three words are the most frequent (22%). Furthermore, 79% of the queries have no more than four words.[1] Each query is a combination of words, names of persons or locations, URLs, special acronyms, program segments, and malicious codes. Some contain words which are very clean, while others may contain typos or meaningless strings which are totally noisy. Some words may just have their meanings as defined in a static dictionary, whereas others may have some special meanings when used on the Internet.

[1]This observation differs slightly from those given in Silverstein et al. [1999] and Jansen [2000], since there are no empty queries within the set of 800,000.

A second difficulty of Web-query classification is that one user query often has multiple meanings. For example, "Apple" can mean a kind of fruit or a computer company. In this case, query classification should provide both meanings, ranked according to the likelihood of each. For "Apple," the result can be given in the form of "Computers\Hardware; Living\Food and Cooking."

A third difficulty is that the meanings of queries may also evolve over time. For example, the word "podcast" can now be interpreted as a kind of Web audio blog-site, but such a word cannot be found in a traditional static dictionary. The distribution of the meanings of this term is therefore a function of time on the Web. In order to preserve the existing meanings of words as well as finding out new ones, we cannot simply classify a query solely based on a static and out-of-date training set [Beitzel et al. 2005]. Instead, we should obtain the meanings of queries from the web in an online manner. Our approach is to retrieve the most related documents for the query, and extract their semantic features. The distribution of the meanings on the Web should influence the ranking of the target categories, among other factors. Also, so as to obtain a better and unbiased understanding of each query, we should not rely on a single search engine, but combine multiple results from different search engines.

In summary, an important issue is how to automatically and effectively classify a large number of queries that are inherently short, noisy, and ambiguous when there is a lack of clear definitions for this data (such as a dictionary) and a lack of additional training data (such as a labeled query-log file). A major traditional approach in handling short and ambiguous queries is through query expansion using relevance feedback [Chang and Hsu 1998], whereby users provide information as to which retrieved result is relevant to the query. However, this does not work for our problem because in many web search problems, the results must be generated automatically and efficiently for the user's. Another major method is query expansion by using a dictionary or thesaurus [Voorhees 1994], whereby the user's query words are enlarged to contain additional information. However, in a web search domain, many queries consist of newly created words and their intended meanings are moving targets. In some sense, generatings a sufficient thesaurus to handle query expansion requires that we solve the query classification problem in the first place. According to our statistics, most queries are short; an illustration is shown for randomly selected queries in Figure 2, where the occurrence frequency of queries is compared to the many number of words in each query. In addition, queries can be noisy as a result of misspellings. A distribution of queries meanings is shown in Figure 3, where we plot the query count in percentage against the number of meanings that each query corresponds to, using the 111 randomly chosen validation samples from the KDDCUP2005 dataset. These 111 queries are labeled by human experts. As can be seen from Figure 3, many queries have more than three possible meanings. All of these characteristics indicate that we cannot solely rely on a static thesaurus to classify them.

Recently, some interesting work has emerged on using query logs to expand the meanings of user queries [Wen et al. 2002; Beeferman and Berger 2000]. However, such a method requires that there exists a query log for us to use, which contains a mapping from submitted queries to clicked Web pages.

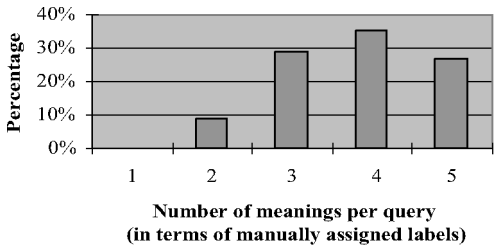


Fig. 3. Percentage of queries with different numbers of meanings (in terms of manually assigned labels).

However, in general, such logs are not available for the timely training and application of query classification models, as in the case of the KDDCUP2005 competition. Even when we can obtain such a log file, the mapping from queries to clicked Web pages does not automatically provide the target categories because the pages themselves still need to be mapped to different categories. In addition, such mapping may be biased by the choice of any single search engine that collects the query logs, and thus the classification approach adopted may not be as general and objective as possible. In order to obtain an objective view of the categories that each query can be mapped to, views from multiple popular search engines should be consulted. Finally, obtaining up-to-date click-through data from multiple search engines raises serious privacy, legal, and business issues, which are beyond the means of many practitioners, businesses, and governments.

This article presents a new approach to classifying large quantities of search queries automatically. Our approach is called *query enrichment*, which takes a short query and classifies it into target categories by making use of a set of intermediate objects. In our application, these intermediate objects are Web pages and category taxonomies such as that of the Open Directory Project (ODP).[2] Query enrichment makes the following basic assumptions:

—Given that the Web is one of the largest data sources available, although we do not know the true meanings of many user queries, their intended meanings should be reflected by the Web as a whole. In addition, although a particular search engine may not fully reflect the intended meanings of the query for the user, when we combine many different search engines and Web directories, the meanings of the query that are embedded in the Web as a whole can be extracted. In other words, by searching the Web for answers, users have expressed their "trust" that their answer is somewhere located on its Web. By this assumption, our approach can be to first "enrich" a query by covering its potential meanings through Web search, and then classify the search results to the target categories.

—A set of objects exist that can "cover" the target categories. An example set of intermediate objects is the ODP, which contains a collection of more than 170,000 different categories and can be taken as an extensive taxonomy.

[2]http://dmoz.com

—We use results from search engines to provide the intermediate objects. Even though a particular search engine may be biased and therefore cannot guarantee high-quality answers to all the search queries from this huge collection, each search engine might provide a different viewpoint in interpreting the user query. The result of different viewpoints should be combined into a coherent whole for better coverage and higher answer robustness.

This last point means that we can submit the queries to multiple search engines, thus the intended answers are among the returned results (i.e., resultant Web pages). In this way, the relative "weights" of multiple search engines in generating collective answers to the queries can be learned from a validation dataset, rather than being predefined.

One possibility is to exploit a metasearch engine. Metasearch engines [Howe and Dreilinger 1997; Selberg and Etzioni 1995] submit queries to multiple different search engines and integrate the results into a single list to be returned to the user. Our task is similar, except that in our case we are interested in query classification rather than search. Therefore, we will follow a similar approach in which we submit the query to different search engines and then classify the query based on the search results from each search engine. In a final step, we integrate the classification results corresponding to each search engine. In machine learning, this is known as an *ensemble of classifiers*. Similar to metasearch engines, the ensemble of classifiers combines the results of different component classifiers using a collection of weights that are learned from a validation dataset. However, as opposed to metasearch engines, the ensemble of classifiers is aimed at classifying a query using a collection of classifiers where each one can be somewhat biased. By integrating the classification results, they compliment each other, and the results are typically more robust than any single classifier. General introductions to ensembles of classifiers are given in Hansen and Salamon [1990], Bauer and Kohavi [1999], Caruana et al. [2004], and Fan et al. [1999].

In general, query enrichment consists of two major steps:

—First, we replace a query by a set of objects wherein the meanings of the query are embedded;

—Second, we classify the query based on the set of objects into ranked target categories.

In order to enrich an input query, in our application, we submit the query to multiple Web search engines and obtain the search results as a ranked list. This ranked list consists of two types of objects: the resultant Web pages and intermediate taxonomies such as the ODP. For example, a query "Apple" submitted to Google (http://www.google.com) returns a list of resultant Web pages that can be mapped to "Computers\Systems; Business\Food_and_ Related_Products" in the ODP directory. The Web pages and the set of categories in intermediate taxonomies combined serve as the basis to map to the target categories.

The rest of the article is as follows. In the next section, we give an overview of our approach. In Sections 3 and 4, we explain the two phases of our solution, respectively. Phase I corresponds to the training phase of machine learning

Table II. Example Queries and Their Categories

Query	Categories
Aerosols	Information\Science & Technology Living\Health & Fitness Living\Family & Kids
Cross pendant	Living\Gifts & Collectables Living\Fashion & Apparel Living\Religion & Belief Shopping\Stores & Products Shopping\Buying Guides & Researching
Aberdeen police department	Information\Law & Politics Information\Local & Regional

algorithms. Phase II corresponds to the testing phase. In Phase I, two kinds of classifiers are developed as base classifiers. One is synonym-based and the other is statistics-based. Phase II consists of two stages. In the first stage, the queries are enriched such that for each query, its related Web pages together with their category information (if they so have) are collected through the use of search engines. In the second stage, the objects in the enriched result are classified through the base classifiers trained in Phase I. Based on the classification results obtained by the base classifiers, we get the last classification results through an ensemble of classifiers. In Section 5, we describe our experimental results on the KDDCUP2005 tasks. We show that through using our solutions we can achieve superior performance as compared to other competitive methods, and similar performance to human labelers when we appropriately integrate search engines and combine query classification results.

2. PROBLEM DEFINITION AND OVERALL APPROACH

The query classification problem is not as well-formed as others such as text classification. The difficulties include short and ambiguous queries and a lack of training data. In this section, we give a formal definition of the query classification problem, which is inspired by the tasks of KDDCUP2005 competition.

Query Classification. The aim of query classification is to classify a user query Q_i into a ranked list of n categories $L_{i1}, L_{i2}, \ldots, L_{in}$, among a set of N categories $\{L_1, L_2, \ldots, L_N\}$. Among the output, L_{i1} is ranked higher than L_{i2}, L_{i2} is higher than L_{i3}, and so on.

The queries are collected from real search engines submitted by Web users. The meanings and intention of the queries are subjective.

The target categories consist of a tree with each node representing a category. The semantic meanings of each category are defined by the labels along the path from root to corresponding node.

In addition, the training data must be found online because in general, labeled training data for query classification is very difficult to obtain.

Table II shows several examples of queries and their categories chosen from test data of the KDDCUP2005 competition. As we can see, each query, may have more than one category, and each category is ordered according to the probabily that the query belongs to it.

To facilitate understanding the definition of query classification as well as the formal discussion, we provide the following definitions.

Definition 1 (*Target Categories*). Target categories are the categories that we will classify user queries into.

For example, for the KDDCUP2005 task, there are 67 categories provided by the organizers as the final targets of query classification (see Appendix A for details).

Definition 2 (*Intermediate Taxonomies*). Associated with a search engine or Web directory, there is often a taxonomy of categories. We wish to distinguish between the target categories in *Definition* 1 and existing taxonomies on the Web. Thus we call the latter "intermediate taxonomies" in this article.

For example, the ODP defines a taxonomy that consists of more than hundreds of thousands of categories organized in a tree structure.

Definition 3 (*Query Enrichment Function*). The query enrichment function u is a function that maps from a query Q to a set of intermediate objects on the web: u: $Q \rightarrow$ intermediate objects. An example of an intermediate object is a Web page. Another type of object is the category label in an intermediate taxonomy such as the ODP directory.

Definition 4 (*Query Classification Function*). A query classification function f_{Q2C} maps from a user query Q to one or more of the target categories: f_{Q2C}: $\mathrm{Q} \rightarrow$ target categories.

Definition 5 (*Text Classification Function*). A text classification function h_{T2C} maps from a body of text T to one or more of the target categories: h_{T2C}: $T \rightarrow$ target categories.

Definition 6 (*Intermediate Taxonomy to Target Category Mapping*). The intermediate taxonomy to target category mapping function l_{IT2C} is a function that maps from a category in the intermediate taxonomy to one or more target categories.

The query classification function f_{Q2C} (*Definition* 4) can be constructed by one of two strategies. The first corresponds to using text-based statistical classifiers. For each query Q:

—We first map the query Q to web pages T;
—We then apply a text classification function h_{T2C} so as to map T to the target categories.

Alternatively, we can build a synonym-based classification function as follows:

—We first submit the query Q to search engines and obtain the category labels of some intermediate taxonomy (these categories differ from target categories).
—We then map the intermediate categories thus obtained to the target categories.

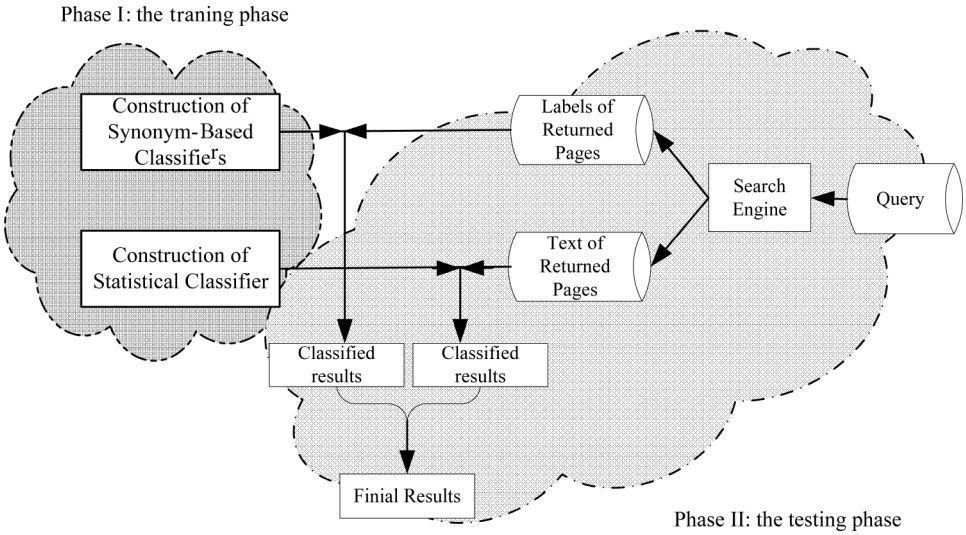


Fig. 4. The architecture of our approach.

By combining the aforementioned two strategies, we can obtain the ensemble-based approach.

As in typical machine learning applications, we adopt two phases in our solution. In Phase I, which corresponds to the training phase of machine learning algorithms, we collect data from the Web for training classifiers that can be used to classify intermediate objects to target categories. In Phase II, corresponding to the testing phase in machine learning research, we apply the classification functions thus built to the target categories. The classifiers learned in Phase I are applied to classify the queries. The overall architecture of our approach is shown in Figure 4 and detailed architectures of the two phases are shown in Figure 5.

For ease of understanding, we will take the KDDCUP2005 task as an example when describing our approach.

3. PHASE I: CLASSIFIER TRAINING

We now discuss Phase I of our approach in detail. In this phase, we train classifiers for mapping from intermediate objects to target categories. As noted, a main problem here is the lack of training data, a difficulty which makes many previous machine learning methods inapplicable. The objective in Phase I is to collect the data from the Web that can be used to train classification functions.

3.1 Synonym-Based Mapping from Intermediate Taxonomies to Target Categories

We first discuss how to construct the mapping functions l_{IT2C} from intermediate to target categories via synonym-based mapping functions. The taxonomies from various search engines can differ both from each other, and from that of the target categories. The mapping function l_{IT2C} can be built by synonym-based

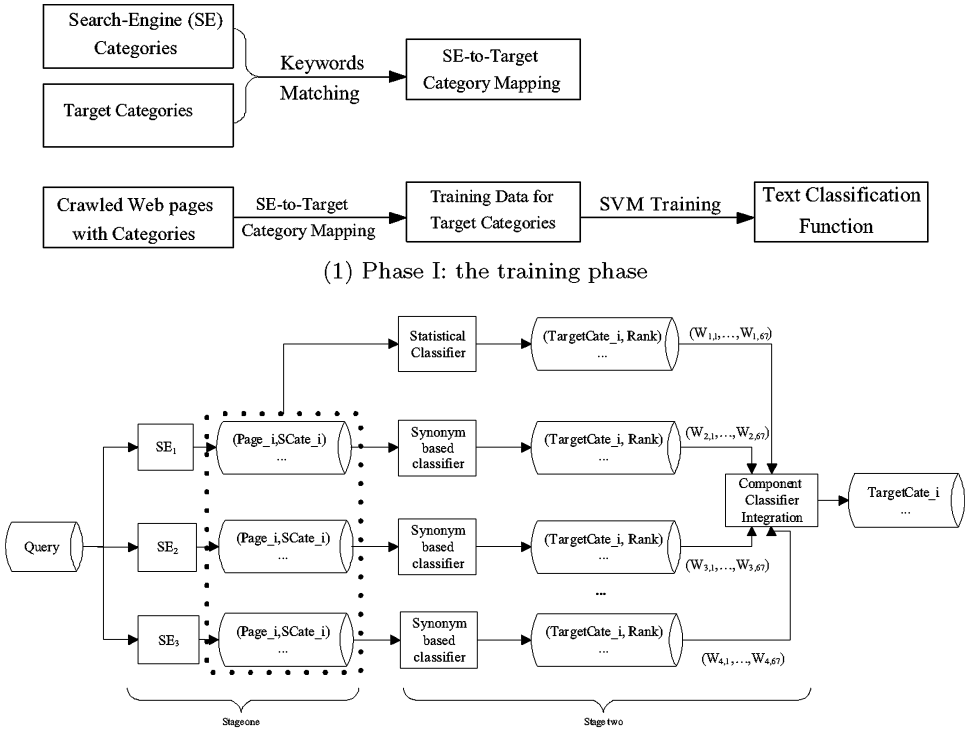


Fig. 5. The two phases: training and applying classifiers.

keyword matching. In this approach, we compare the terms used to describe the target categories with those in the intermediate taxonomies.

Consider two categories, c_1 and c_2, where c_1 is a category label from the target categories and c_2 is a label from an intermediate taxonomy. If they share some of the same keywords, we can map c_2 to c_1 directly. For example, in the KDDCUP2005 task, the target categories have two levels, for example, "Computers\Hardware." The second level specifies a particular field within the first level. For most target categories, we only consider keywords in the second level because they cannot be confused with other categories. A typical example is "Computers\Internet and Intranet." Even if we do not consider the first level, "Computers," there are no other categories which may be confused with "Internet and Intranet." Therefore, if a category from an intermediate taxonomy contains either "Internet" or "Intranet," we will map it to "Computers\Internet and Intranet." However, many categories are more difficult to deal with. For these, we require that the keywords in the first and second levels match the categories in the intermediate taxonomy simultaneously. Otherwise, we cannot distinguish between two categories that share the same keywords only in the second level, such as "Computers\Hardware" and "Living\Tools and Hardware." Although both have the keyword "Hardware" in the second level, they

belong to the two different domains "Computers" and "Living," as defined by the first level.

In order to improve the coverage of the mapping function l_{IT2C}, we can in advance extend the keywords in the category names to include both singular and plural forms. For example, "Living\Book and Magazine" is extended to "Living\Book and Magazine and Books and Magazines."

After applying the preceding procedure, we may still miss a large number of mappings. Many categories in the intermediate taxonomy do not occur in target categories, although they share words with the same meanings. In response, we expand the keywords in each label in the target categories through WordNet [Miller et al. 1990] . For example, the keyword "Hardware" is extended to "Hardware and Devices and Equipments" and the keyword "Movies" is extended to "Movies and Films."

3.2 Text Data Collection and Statistical Classifier Training

We now discuss how to build the statistical classification function h_{T2C} to map from a body of text T to a target category. As we will discuss in Section 4.2, synonym-based classifiers have a certain drawback—they have low recall. In order to address this problem, we consider statistical classifiers to help classify queries. A statistical classifier classifies a query based on the semantic content of a text, which can provide better recall as well as precision. Even when the synonym-based classifier fails to give any meaningful mapping for a query, the query can still be classified by a statistical classifier. Any kind of statistical text classifiers, such as naive Bayes [McCallum and Nigam 1998], KNN [Yang 1999], and support vector machine (SVM) [Joachims 1998, 1999], can be applied.

To construct a statistical classifier, the first step is to collect training data. This step is nontrivial, since no training data is provided explicitly, as we have stated. In order to collect the training data, we use the following methods:

—First, we collect Web pages from some online manually labeled Web page directories, such as the ODP.
—By applying function l_{IT2C}, we can map a collected web page into the target categories. Thus, the mapped Web pages can be used as the training document for target categories.
—The training data among the target categories is usually extremely unbalanced. In order to remove the impact of unbalanced distributions, as well as speed-up the training step, we need to sample the training documents.

After training examples are collected for each target category, we can follow the traditional procedure to train statistical classifiers, including some proper preprocessing steps and parameter tuning.

To clarify the previous procedure, we illustrate it through our solution to the KDDCUP2005 task. We use the SVM classifier because of its high generalization performance when used for text classification tasks and its easy implementation. About 1,540,000 Web pages are collected from ODP in total. After applying the mapping function between the ODP and KDDCUP2005 categories, only 500,000 Web pages fall into KDDCUP2005 categories. To

address the unbalanced distribution of Web pages among the 67 categories, we randomly selected 15,000 Web pages from those categories containing more than 15,000 Web pages and keep all the pages for the categories with less than 15,000. Document frequency (DF) and information gain (IG) methods are used for feature selection [Yang and Pedersen 1997]. Then we use the SVM^{light} software package (http://svmlight.joachims.org/) to train an SVM. A linear kernel is used and the one-against-the-rest approach is applied for the multiclass case.

4. PHASE II: QUERY CLASSIFICATION

Phase I of our algorithm is designed to collect data for training the mapping functions, which include synonym- and statistics-based classifiers. Phase II of our algorithm is devoted to subsequently classifying a query to one or more target categories based on the classifiers.

Phase II is conducted in two stages. The first is to enrich the queries by searching those related pages that can specify the meanings of the queries most accurately. Enrichment is necessary because the queries are rather short, and their meanings ambiguous. We perform the enrichment process by finding relevant text from related Web pages, as well as the category information of these Web pages (if they so have) through Web search.

The second stage is to classify the queries based on the data collected in the first stage and classifiers trained in Phase I. In this stage, we takes the two kinds of classifiers with totally different mechanisms that were trained in Phase I as the base classifiers and develop several ensemble classifiers with different ensemble strategies to classify the queries. Experimental results show that ensemble classifiers can improve the classification performance significantly.

4.1 Query Enrichment

Query enrichment is a key step because our goal is to classify short and ambiguous queries, without any additional descriptions about them. After this step, two kinds of information for each query are collected. One is the list of Web pages related to the target query. The other is the set of categories corresponding to the related pages. These two kinds of information will be leveraged by the two kinds of classifiers trained in Phase I, respectively.

In our approach, we send each query to multiple search engines that can provide options for both directory and Web search. Directory search in a search engine refers to search algorithms that return the related pages of a query, together with the page's categories. Since these categories of Web pages are labeled by humans, it is appropriate to use them to classify the queries. However, not all the pages indexed by the search algorithm contain category information; in this case, Web search can return more related pages than directory search. Based on the contents of the pages returned by Web search, we can classify the queries using a text classification algorithm.

In summary, to enrich a query through search engines, we use the following three steps:

(1) We first try to get the related pages through directory search;
(2) If we cannot get any results from Step 1, we try a Web search;

(3) If we still cannot get any results, the queries must either be too noisy or totally meaningless. Thus, we use some preprocessing approaches to clean them up and then resubmit them to directory and Web search, in turn, as done in Steps 1 and 2. If, after this step, there is still no result returned, the queries will not be processed any further, and no classification results are generated. Currently, two preprocessing approaches are employed for the cleaning-up, which will be described in detail in the next example.

In our solution to the KDDCUP2005 task, we use three search engines, including Google, Looksmart, and a search engine developed by ourselves based on Lemur.[3]

Now, let us use Google as an example to illustrate the three steps in detail.

—Initially, all 800,000 queries are preprocessed by removing special characters such as ",#,%," while keeping the letters and digits. Then these 800,000 queries are sent to the Google directory search. We are able to retrieve related pages for about 500,000 (63%) queries in this way.

—We then send the remaining 300,000 queries to the Google Web search and get results for an additional 200,000 queries.

—For the remaining 100,000 queries, we conduct further preprocessing. We use the function of "Did you mean," provided by Google, to trigger a search for queries that are the most relevant to the originals. For example, given the query "a cantamoeba," neither Google directory search nor Web search returned any results. However, by trying "Did you mean," Google could return results for the word "acanthamoeba," which is related to health, disease, and medicine. In this way, we can get the results for another set of 60,000 queries from the Google directory search or Web search.

—However, after this step, there are still 40,000 queries left without any results. These are very noisy. They usually consist of connected words without spaces, long, meaningless clobbers, or URL addresses containing parameters or even malicious codes. We try to render these queries meaningful by extracting words from them through a maximum-length matching method based on the WordNet dictionary. This method tries to extract as many meaningful words as possible and to make each as long as possible. Taking the query "wheelbearingproblemsdamage" as an example, Google cannot return any results through either a directory or Web search or even, "Did you mean" a function. Therefore, we can split the whole query into the four meaningful words "wheel bearing problems damage." After doing this, we can get reasonable results from the Google directory or Web search. In this way, we can get the results for 30,000 of the remaining 40,000 noisy queries.

—The remaining 10,000 queries cannot receive any pages from Google as answers. These queries are inherently noisy and meaningless, such as "dddddfdfdfdfdf." Therefore, our classifier will not return any answers for these queries, although we can assign labels to them according to the prior

[3]http://www.lemurproject.org/

distribution of the target category. This potentially reduces recall (one measurement of classification performance) for the query classifier. But because these outliers only correspond to a tiny portion of all queries (1.25%), they do not have much effect on the final classification results.

We follow the same steps when using Looksmart. Among the 800,000 queries, about 200,000 have directory search results. Another 400,000 have Web search results, and the remaining 200,000 have none.

The third engine we use was developed by ourselves based on Lemur. We first crawled more than one million Web pages with the category information from ODP. Then we indexed this collection of pages with Lemur. Given a query, Lemur can retrieve a number of pages that are most relevant, together with their corresponding categories. Therefore, the function of this Lemur-based search engine and the ODP data is similar to the directory search provided by Google and Looksmart. Using this search engine, we can retrieve related pages for most of the 800,000 queries (except 35,000).

In summary, after enriching queries through a search engine, we can obtain two lists for each query. One is the returned Web pages list from a search engine, and the other is a category list attached to the Web pages in the Web page list. Note that some Web pages have no category information. These two lists will be leveraged by different kinds of classifiers individually.

4.2 Query Classification Using Synonym-Based Mapping Functions

Using the synonym-based category mapping functions discussed in Section 3.1, we can now build a query-to-target-category mapping function f_{Q2C}. In particular, for each query Q, through the function u constructed in Section 4.1, we can obtain a set of intermediate categories S of the related Web pages returned by a given search engine. Then we apply the category-mapping function l_{IT2C} to S, which returns an ordered list of target categories.

Using different search engines, we might obtain different intermediate categories. For each category of a given search engine's taxonomy, we can construct a mapping function l_{IT2C} between it and the target categories, as shown in Section 3.1. After obtaining these mapping functions, we can perform query classification based on the intermediate categories S returned by the search engine. We then map intermediate to target categories using the constructed mapping functions. We keep track of the number of times each target category is mapped onto. We can then obtain the target categories, together with their occurrence frequencies. By ranking categories in terms of occurrence frequencies, we get a ranked list of final target categories into which the query can be classified. Based on the various intermediate category lists and their corresponding mapping functions, we obtain different classification results. The classification functions thus obtained are known as synonym-based classifiers.

Based on the aforementioned approach, synonym-based classifiers tend to produce results with high precision and low recall. They produce high precision because they are based on manually annotated Web pages and can utilize the classification knowledge of human editors. Therefore, once a mapping function is constructed, the classification result is highly probable to be correct.

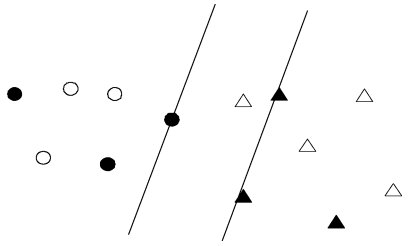


Fig. 6. Illustration of the advantage of statistical classifiers.

For example, we have shown that categories in the intermediate taxonomy, such as "...\Computers\...\ Hardware\...," are mapped to the target category "Computers\Hardware." Therefore, once the Web pages associated with a given query fall into the category "Computers\Hardware\Storage\Hard Drives," we can assign "Computers\Hardware" to the query with high confidence. Synonym-based classifiers produce low recall because it is hard to find all the mappings from the intermediate taxonomy to the target categories by keyword mapping. For example, about 80,000 out of 354,000 categories in Google's taxonomy are not mapped onto target categories. Therefore, we cannot map all the intermediate categories in the category list for a given query to the 67 target categories, and may miss some correct categories for the query.

Another reason for the low recall problem is that a search engine may return only a few or even no Web pages with categories. The synonym-based classifier may fail because of the search results shortage problem. Therefore, we need to construct other classifiers to help handle the low recall problem, which we will describe in the next section.

4.3 Query Classification Using Statistical Classifiers

In this section we discuss the use of statistical classifiers to classify queries. As shown in Section 4.1, after submitting a query Q to a search engine, we get a list of Web pages. Then, we can extract a body of text from the Web pages to capture the occurrence context of the issued query, which can help determine its potential meanings. To accomplish this, we keep the top N results in the returned list (the parameter N will be studied in Section 5) and use the aggregate terms of the corresponding snippets, titles, and URL terms to represent the query. The query's bag of terms will be processed by stop-word removing, stemming, and feature selection. The resultant term vector can be used as input for the statistical classifiers and a ranked list of categories for each query will be produced.

Statistical classifiers are expected to achieve a higher recall than that of synonym-based classifiers. The reason is illustrated in Figure 6. For simplicity, only two categories are considered. The circles and triangles represent samples belonging to the two categories. The black symbols represent the training data we collect for the two categories through the means shown in Section 3.2. The white symbols represent the Web pages that we crawled from the ODP which should have been mapped to the target categories, but were not because of

the drawbacks of mapping function l_{IT2C} from the intermediate taxonomy to the target categories. Given a query Q, after the query enrichment step, we may happen to get only the white circles or triangles to represent the query. Thus, it is clear that we cannot judge the labels of the query by synonym-based classifiers, since there is no mapping relationship between the white circles and triangles to the target categories. However, statistical classifiers can still obtain the separating hyperplane based on the black circles and triangles, which can be used to classify the white, and hence can further be used for classifying the query Q.

4.4 Ensemble of Classifiers

From the previous two approaches, we can independently build various classifiers that can classify input queries into target categories. These approaches are based on different mechanisms and can be complementary to each other. Previous work has shown that the proper combination of different base classifiers can improve final classification performance [Hansen and Salamon 1990; Kittler et al. 1998; Bauer and Kohavi 1999; Cann 2003; Fan et al. 1999]. In this section, we consider how to combine them.

Dietterich [2000] and Alpaydin [2004] have categorized different ensemble-based methods. Of these, voting, bagging, and boosting are the major ones. In voting, which defines a linear combination of existing classifiers, the main task is to decide the weights used in the combination. In bagging, the base classifiers are generated to differ by training them over slightly different training data that is sampled from the given training set by bootstrap. In boosting, the base classifiers are generated sequentially, where each is trained on the mistakes of the previous. Both bagging and boosting require that there is sufficient labeled data for generating base classifiers [Meyer and Brown 1998; Dietterich 2000]. Because in query classification, labeled data is very scarce (in our case, there are only 111, which corresponds to 0.014% of all the data), voting becomes the most natural choice. We thus focus on how to set the weights of different classifiers.

Figure 7 illustrates our ensemble-based method for query classification. As we can see from the figure, the results of the classifiers are linearly combined to generate the final category ranking. In our approach, we consider two ways to assign the combination weights for different classifiers. The first is to make use of the validation dataset involving a small number of queries labeled by humans. Considering that the validation dataset is too small and easily overfitting, an alternative is to ignore this validation dataset and instead give each base classifier equal weight. More details about the first strategy are shown to follow.

As we can imagine, the different classifiers introduced in the preceding sections have differing performance. Some may work better than others on certain categories. For example, a classifier may achieve high precision on one category, while having high recall on another. This indicates that it is not proper to assign a single weight to a classifier. Instead, we should differentiate the weight of a classifier on different categories, according to its performance. To determine the weights, we validate each base classifier on the validation samples. The

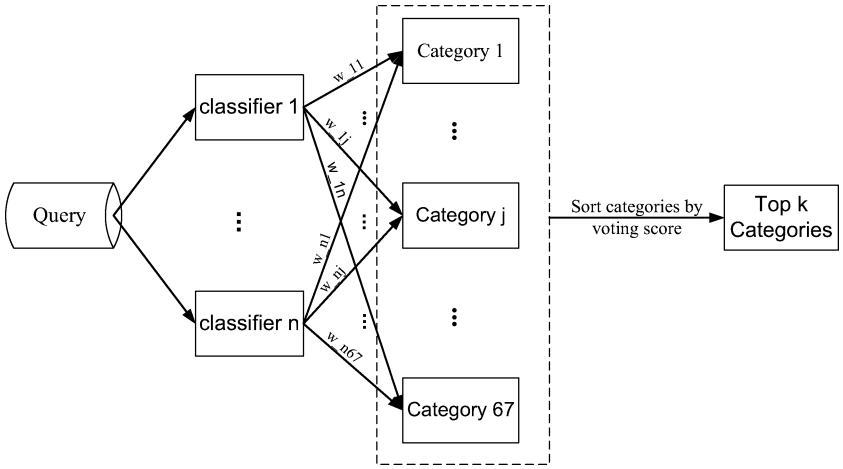


Fig. 7. Illustration of the ensemble classifiers.

higher precision a classifier achieves on a given category, the higher the weight assigned to it classifier on this category. As a result, each classifier may obtain a weight value W_{ij} on a target category j. W_{ij} is defined by:

$$W_{ij} = \frac{p_{ij}}{\sum_{k=1..n} p_{kj}},$$

where p_{ij} is the precision of classifier i on categories j. The definition of precision will be given in Section 5.2.

Three additional ways to determine combination weights, similar to the previous, will also be discussed and tested in Section 5.

5. EXPERIMENTS AND DISCUSSIONS

To test our proposed approach, we conduct extensive experiments on the KDDCUP2005 datasets. The experimental results validate the effectiveness of our approach. In addition, we give an analysis of the consistency of the three labelers on their judgments of classifier performance.

As introduced in the preceding section, we now have six classifiers in total of three synonym-based, one statistical SVM and two ensemble. For simplicity, we refer to the three synonym-based classifiers that rely on Google, Looksmart, and Lemur as S1, S2, and S3, respectively. We denote the ensemble classifier that relies on the validation dataset as EDP (since it assigns different weights for each base classifier on (D)ifferent categories according to its (P)recision on the category) and the one that does (Not) as EN.

5.1 Datasets

One of the KDDCUP2005 datasets contains 111 sample queries, together with human labeled categories. These samples help exemplify the format of the queries and provide the semantics for a tiny number of them. In fact, since the category information of these queries is truthful, they can serve as the

validation data for our proposed classifiers. Another dataset provided by the organizers contains 800,000 queries in total, which are selected from MSN search logs to test the submitted solutions. Since manually labeling all 800,000 queries is too expensive and time-consuming, the organizers at last randomly selected 800 and invited three human labelers to label them. We denote the three labelers (and sometimes the datasets labeled by them, if no confusion is caused) as L1, L2, and L3, respectively. We refer to the former as the *sample* dataset and the latter as the *testing* dataset in the following sections. Both of these datasets can be used to evaluate the different classification approaches. The sample dataset can in addition be used to determine the ensemble weights in this article.

5.2 Evaluation Criteria

The evaluation criteria adopted by the KDDCUP2005 organizers are the standard measures for evaluating classification performance in information retrieval (IR), including precision, recall, and the F1-measure [Van 1979]. The definitions of precision, recall and F1 in query classification context are given as follows:

$$
\begin{aligned}
&A: \sum_i \#\ \textit{of queries correctly tagged as}\ c_i \\
&B: \sum_i \#\ \textit{of queries tagged as}\ c_i \\
&C: \sum_i \#\ \textit{of queries whose category is}\ c_i \\
&\textit{Precision} = \tfrac{A}{B} \\
&\textit{Recall} = \tfrac{A}{C} \\
&F1 = \tfrac{2 \times \textit{Precision} \times \textit{Recall}}{\textit{Presion} + \textit{Recall}}
\end{aligned}
$$

For the sample dataset, we report the precision, recall, and F1 evaluation results for each classifier. For the testing dataset, since the three labelers are asked to label the queries, the results reported are the average values [Li et al. 2005]. Take the calculation of F1 as an example:

$$\text{Overall F1} = \frac{1}{3}\sum_{i=1}^{3}(\text{F1 against human labeler } i)$$

For some of the ensemble classifiers, we need to know the performance of a classifier on a certain category. It is easy to define such criteria according to the preceding definition. For example, the precision of classifier i on category j could be defined as:

$$P_{ij} = \frac{\#\ \textit{of queries are correctly tagged as}\ c_j\ \textit{by classifier}\ i}{\#\ \textit{of queries are tagged as}\ c_j\ \textit{by classifier}\ i}.$$

Because there are 67 target categories, a random classification algorithm would give a precision of $1/67 = 1.5\%$ if one category is to be returned, and $5/67 = 7.5\%$ if five are returned. Therefore, we are comparing against a baseline of 7.5% for the KDDCUP2005 task, whose requirement is to return, at most, five results.

Table III. Number of Queries with N Labels

N	L1	L2	L3
1	14	118	16
2	138	365	79
3	186	225	212
4	222	71	199
5	240	21	294
Ave	3.67	2.39	3.845

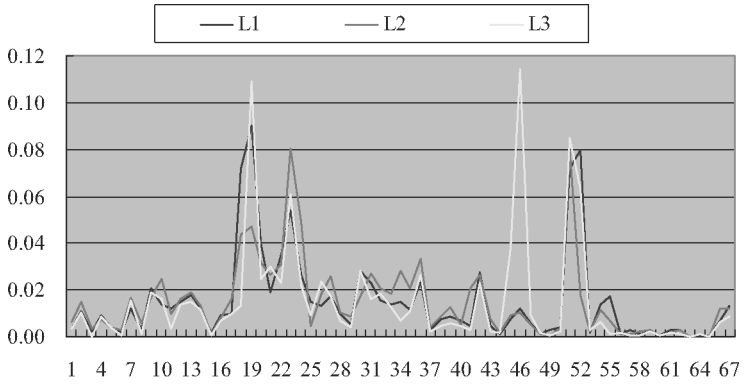


Fig. 8. The distribution of labels assigned by the three labelers.

5.3 Quality of the Testing Dataset

Because the testing dataset is provided by three human labelers, we wish to evaluate its quality. In particular, in this section, we analyze the consistency between the three labelers for the testing dataset.

Table III gives the distribution of the number of categories assigned by human labelers to each query, that is, how many queries are assigned N categories, where N changes from one to five. The "Ave" row shows the average number of categories for each query. From the table, it seems that the three labelers disagree to quite an extent. However, from Figure 8, which shows the distribution of the 67 categories assigned by the three labelers to 800 testing queries, we can see that the distributions for the three human labelers are very similar. From Table III and Figure 8, we can conclude that the general distributions of categories are very similar, though the labelers L1 and L3 tend to assign more categories for each query than does L2.

Figure 9 shows the F1 values of the six classifiers on testing data labeled by the three labelers. From the figure, we can see that the labelers have a high correlation with respect to the relative performance of the classifiers, especially L1 and L3. After ranking the six classifiers according to each labeler, we calculate the Spearman rank-order correlation coefficient between each pair of labelers [Hoel 1966]. Spearman correlation is a nonparametric approach to calculating the relationship between two variables based on rank and no assumption about the distribution of values is made. The results are shown in Table IV.

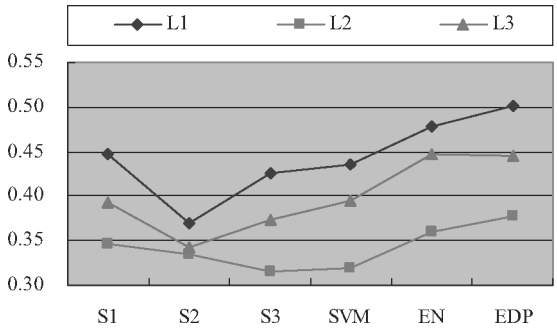


Fig. 9. Performance of various classifiers evaluated by different labelers.

Table IV. Spearman Correlation Between Each Pair of Labelers

L1.vs.L2	L1.vs.L3	L2.vs.L3
0.829	0.943	0.771

Table V. Performance of Each Labeler Against Another

(1) Precision

	L1	L2	L3
L1	1.000	0.637	0.561
L2	0.415	1.000	0.367
L3	0.588	0.590	1.000

(2) Recall

L1	1.000	0.415	0.588
L2	0.637	1.000	0.590
L3	0.561	0.367	1.000

(3) F1

	L1	L2	L3
L1	1.000	0.502	0.574
L2	0.502	1.000	0.452
L3	0.574	0.452	1.000

In general terms, correlation coefficients of over 0.67 indicate strong relationships [Cann 2003]. So, we can conclude that the three labelers are highly correlated when they determine the performance of classifiers.

Besides the aforementioned correlation analysis, we also investigate the performance of each labeler when taking the other two labelers as the ground truth. The results are shown in Table V. The labelers in columns are tested against labelers in rows (which are taken as the ground truth). The average F1 value of the labelers is 0.509. Therefore, we can conclude that the query classification problem is not easy and the performance of our proposed ensemble classifier (with F1 equal to 0.444) is close to the average performance of the three human labelers.

Table VI. Summary of the Classifiers

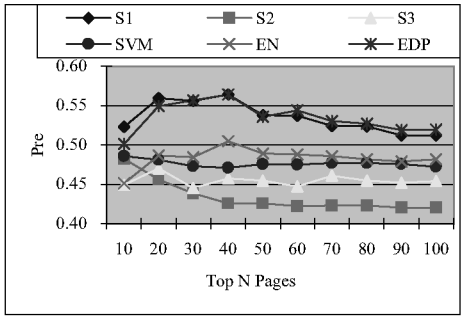
Symbols of Classifiers	Meaning of the Classifier
S1	Synonym-based classifier based on Google
S2	Synonym-based classifier based on Looksmart
S3	Synonym-based classifier based on Lemur
EDP	*E*nsemble classifier in which the weights for each base classifier on *D*ifferent categories are set in proportion to its *P*recision on the category when tested on the validation dataset
EDF	*E*nsemble classifier in which the weights for each base classifier on *D*ifferent categories are set in proportion to its *F*1 on the category when tested on the validation dataset
EP	*E*nsemble classifier in which each base classifier is assigned a single weight in proportion to its overall *P*recision when tested on the validation dataset
EF	*E*nsemble classifier in which each base classifier is assigned a single weight in proportion to its overall *F*1 when tested on the validation dataset
EN	*E*nsemble classifier in which each base classifier is equally weighted and does *N*ot rely on the validation dataset

5.4 Experimental Results and Explanation

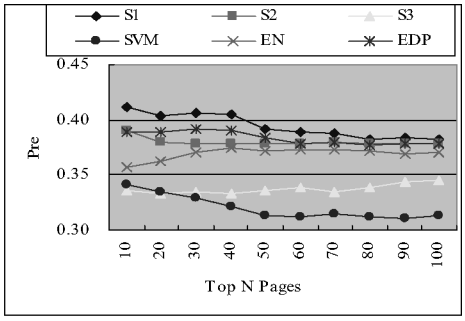
In the following, we first investigate the two main parameters affecting the performance of our proposed classifiers on the two datasets, and we then compare the performance of these classifiers. Afterwards, we investigate several other ensemble strategies, besides the two we introduced in Section 4.4. Finally, we compare our approaches with those of other participants. Table VI shows a summary of the compared classifiers, including the six we have introduced and three additional ensemble classifiers we will introduce in Section 5.4.3.

5.4.1 *Effect of Parameter Tuning.* There are two main parameters that significantly impact the performance of our proposed classifiers. The first is the number of result pages returned by the search engines, which we should use for enriching each query. If we use too few pages, we may fail to cover its diverse topics. However, if we use too many, we may introduce a great deal of noise. Figure 10 shows the performance of different classifiers with respect to the increasing number of related pages used for classification. Here, related pages are considered in the order in which they are returned by the search engines, that is, in the order of degree of relevance with the query. The results shown in the figure verify our conjecture. As the number of related pages increases, the precision increases initially, and then tends to decrease. The reason is that we need a certain amount of pages to get the meaning of the query. However, if we include too many, noise may be introduced, which can reduce the precision. From Figure 10, we can see that the critical point for most classifiers is 40 pages. Before this, classifier performance increases as we use more pages, while after this point, the performance begins to decrease. Therefore, in the following experiments we keep only the top 40 pages for query classification.

Another parameter is the number of labels we should assign to each query. The KDDCUP2005 competition rules allow us to assign, at most, five labels for each query. However, in order to achieve higher precision, we should assign

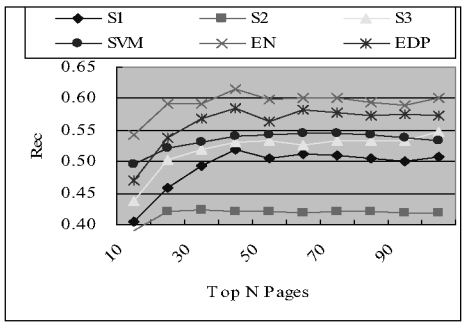


(a) precision on sample dataset

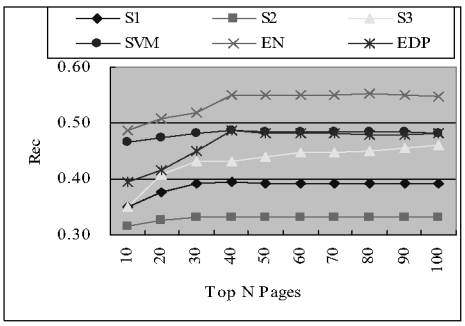


(b) precision on testing dataset

(1) Precision of the six classifiers.

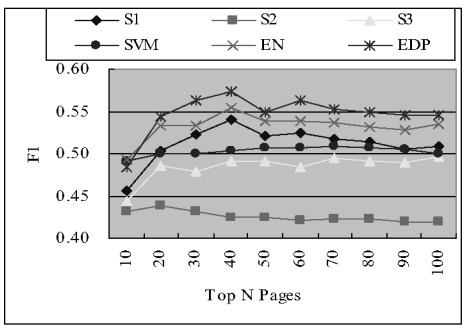


(c) recall on sample dataset

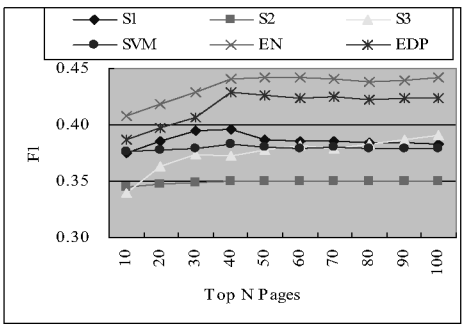


(d) recall on testing dataset

(2) Recall of the six classifiers.



(e)F1 on sample dataset



(f) F1 on testing dataset

(3) F1 of the six classifiers.

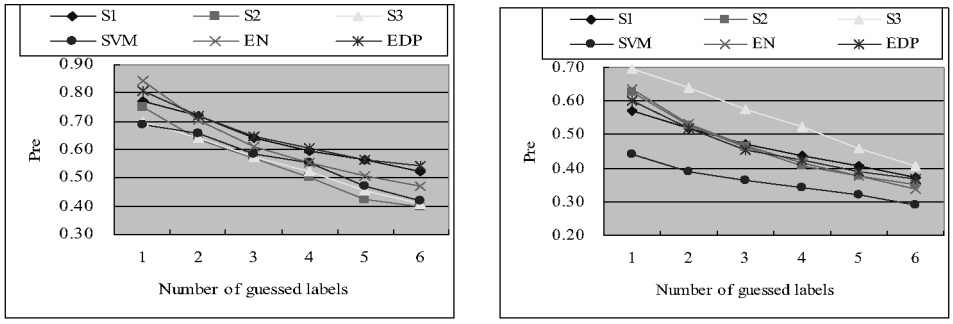
Fig. 10. Performances of different classifiers vary with the number of used related pages on the two datasets.

few, but accurate labels, whereas if we hope to achieve higher recall, we need to assign more possible labels. Figure 11 shows the performance of different classifiers by varying the number of classified categories. As we expected, as the number of classified categories increases, the precision of all the classifiers decreases, while recall increases significantly. In contrast, the value of F1 increases initially, and then decreases. For most of the classifiers, maximum values of F1 are achieved when four categories are generated for each query. Although the F1 values are close to those obtained when five categories are assigned, the precision values are much lower. We also adopted some heuristic rules to decide the number of classified categories. As shown earlier, for each query, our classifiers can return a ranked category list according to a certain criterion, which is referred to as confidence. Among the top five categories, if the confidences of two neighboring categories c_i and c_{i+1} vary too much, we stop at c_i and discard the categories after c_i. This heuristic rule can help us to some extent, but not greatly.

5.4.2 *Comparison Between Classifiers.* From the previously described experimental results on both the sample and testing datasets, we can see that of the four base classifiers, S1 works best while S2 works most poorly. The lack of overlap among the search results from different search engines explains the performance differences among S1, S2, and S3. The reason S2 does not work well is that for many queries, we cannot obtain enough related pages through the Looksmart search engine. For the SVM, we expect that it can solve the low recall problem caused by the three synonym-based classifiers, as discussed in Section 4. Figures 10 and 11 show that the SVM does obtain the highest recall in most cases, as compared to synonym-based classifiers. We also notice that the two ensemble classifiers can achieve better performance in terms of F1 than any other base classifier. For the peak F1 values of the three best classifiers on the testing dataset (EN, EDP, and S1), we can see that, compared with S1, EN and EDP improve the F1 measure by 12.1% and 8.3%, respectively. In fact, when we design these two ensemble classifiers, EDP is expected to achieve higher precision because each component classifier is highly weighted on the categories in which it achieves high precision, and EN is expected to achieve higher F1 performance, since the recall is relatively high. According to our two submitted results, the F1 value of EN (0.444) achieves a 4.2% relative improvement compared with that of EDP (0.426). However, the precision value of EDP (0.424) improves by 2.3%, while that of EN is 0.414.

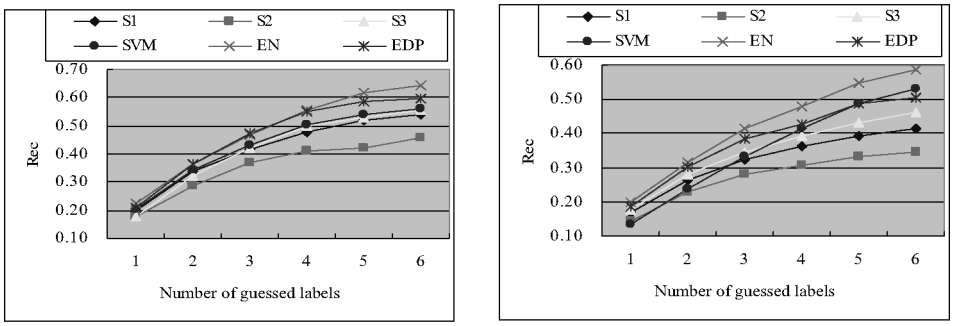
5.4.3 *Study of Different Ensemble Strategies and the Effect of the Validation Dataset.* As shown earlier, we employed two strategies to decide the weights for ensemble classifiers. In this section, we will study three more strategies. Besides this, we also test the effect of validation data. The strategies we will study are shown next. For clarity, the strategy EDP which was introduced in Section 4.4 is repeated here.

—The weight for classifier i on category j is in proportion to P_{ij}. P_{ij} refers to the precision of classifier i on category j. We denote this strategy as EDP.



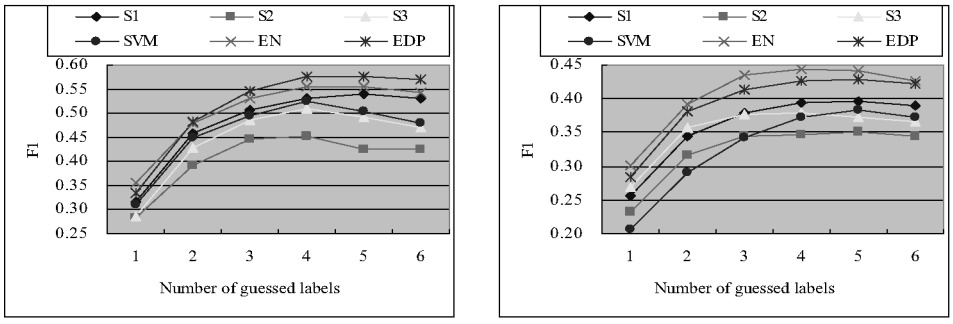
(a) precision on sample dataset (b) precision on testing dataset

(1) Precision of the six classifiers.



(c) recall on sample dataset (d) recall on testing dataset

(2) Recall of the six classifiers.



(e) F1 on sample dataset (f) F1 on testing dataset

(3) F1 of the six classifiers.

Fig. 11. Performances of different classifiers vary with the number of classified categories on the two datasets.

Table VII. Comparison Between Different Ensemble Strategies

(1) Precision

	EDP	EDF	EP	EF
20%	0.375	0.364	0.389	0.385EF
40%	0.381	0.372	0.388	0.387EF
60%	0.386	0.376	0.387	0.386EF
80%	0.388	0.378	0.387	0.386EF
100%	0.390	0.380	0.387	0.386EF

(2) Recall

	EDP	EDF	EP	EF
20%	0.419	0.422	0.503	0.500
40%	0.453	0.456	0.500	0.500
60%	0.478	0.482	0.500	0.500
80%	0.483	0.488	0.500	0.500
100%	0.488	0.492	0.500	0.500

(3) F1

	EDP	EDF	EP	EF
20%	0.393	0.388	0.435	0.431
40%	0.410	0.406	0.433	0.432
60%	0.424	0.419	0.433	0.432
80%	0.425	0.422	0.433	0.432
100%	0.429	0.425	0.433	0.432

—The weight for classifier i on category j is in proportion to $F1_{ij}$. $F1_{ij}$ refers to the F1 value of classifier i on category j, where $F1_{ij}$ can be defined in a similar way as P_{ij}. We denote it as EDF.

—Each classifier is assigned a single weight in proportion to its overall precision. We denote this as EP.

—Each classifier is assigned a single weight in proportion to its overall F1. We denote it as EF.

EF and EP differ from EDP and EDF in that we assign a unique weight for a base classifier across all categories in EP and EF, while we assign each base classifier different weights on the various categories in EDP and EDF.

In order for the four strategies to determine the weights, we rely on the validation data (the 111 samples). The size of the validation data may affect the performance of different strategies. To test its effect, we construct five subsets by randomly picking 20%, 40%, 60%, 80%, and 100% samples. Then we construct the ensemble classifiers based on these subsets, respectively. In order to remove the variance, we conduct the experiments three times. The final reported values have been averaged across these three runs.

From Table VII, we can see that with increasing increment of size of the validation dataset, the performance of EDP and EDF increases steadily. When the validation dataset is small, the performance of a classifier on a given category may not reflect the real performance of the classifier. Therefore, the weights generated tend to be unreliable and cause poor performance of the ensemble

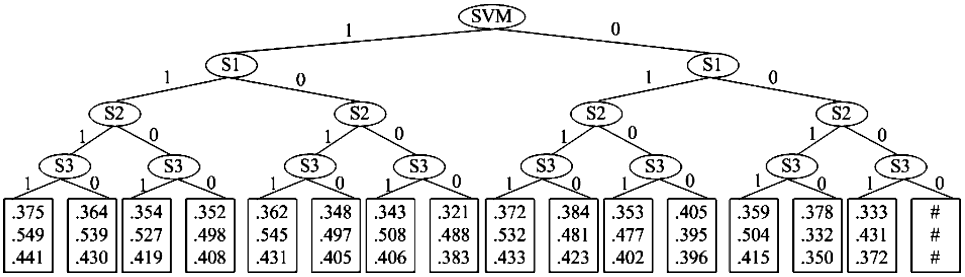


Fig. 12. Performance of all ensemble classifiers constructed from the four base classifiers.

classifiers. However, with increasing size of the validation dataset, the performance of the base classifiers becomes more and more reliable, as do the weights we obtain. Consequently, the performance of the ensemble classifiers improves. Note that the performance of EP and EF does not change much with the change of size of the validation dataset. By considering the performance of EN, which assigns equal weights to each base classifier, we can obtain a reasonable explanation. As we can see, the performance among different base classifiers does not vary too much on a given dataset in terms of precision and F1. Therefore, the weights for each base classifier generated according to EP and EF are similar. That is, EP and EF both perform similar to EN, regardless of which validation dataset is used. Hence, the performance of EP and EF does not fluctuate much with change of the validation dataset.

From Table VII, we also find that EDP (EP) always achieves better precision than EDF (EF) across different validation datasets, without sacrificing the value of F1. We can conclude that in order to achieve higher precision from ensemble classifiers, we should determine the weights of base classifiers according to their precision values. We can see that the precision achieved by EDP increases steadily, whereas that achieved by EP is relatively stable, with increase of the validation dataset. When 80% or more of the 111 samples are used, EDP outperforms EP in terms of precision, although the advantage of EDP is not obvious. Therefore, in order to reach higher precision, it is necessary to try EDP if we have a large validation dataset.

5.4.4 *Effect of Base Classifiers.* In the preceding section, we studied the effects of different ensemble strategies, as well as that of the validation dataset. In this section, we will study the effect of the number of base classifiers. For simplicity, we assign each base classifier equal weight when constructing an ensemble classifier, as with EN. Given four base classifiers, we can obtain 15 ensemble classifiers in total.

Figure 12 shows the performance of all the ensemble classifiers on the testing dataset. Each leaf in the tree represents an ensemble classifier. On the path from a leaf to the root, "1" indicates that the corresponding base classifier is included to construct the target ensemble classifier; "0" indicates the opposite case. The three numbers in each leaf reflect the performance of the corresponding ensemble classifier in terms of precision, recall, and F-measure. From the figure, we can conclude that the more base classifiers included, the better the

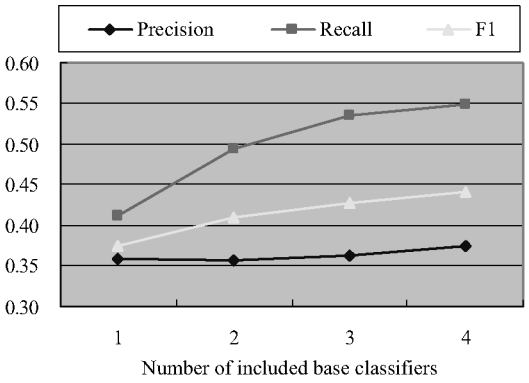


Fig. 13. Averaged performance of different kinds of ensemble classifiers categorized according to number of base classifiers.

Table VIII. Average Running-Time of Each Step for Testing a Query (seconds)

Retrieve Pages from Search Engines			Process	Classify through Classifiers	
Google	Looksmart	Lemur + ODP	Returned Pages	Synonym-Based	SVM
0.98s	0.39s	0.11s	0.002s	0.0006s	0.0035s

performance. Figure 13 further clarifies the conclusion. In fact, by categorizing the 15 ensemble classifiers according to number of base classifiers included, we acquire four kinds. Figure 13 is obtained by averaging the different ensemble classifiers within each type. From Figure 12, we also find that whenever the base classifier SVM is included, there will be an obvious improvement in recall, while precision does not change much, and thus F1 is improved. This observation once again validates our idea that the two kinds of base classifiers complement each other and the combination can improve the classification results.

5.4.5 *Running-Time Analysis.* In this section, we analyze the time complexity of our approaches. Since the training stage of our approaches can be completed offline, we only consider the test stage. As shown before, to test a query, we need four steps: (1) submit the query to search engines and fetch the related pages; (2) process the returned pages to obtain the intermediate category information for each page and textual representation of the query; (3) apply the two kinds of base classifiers on the enriched representation of the query; and (4) combine the results of the base classifiers. Table VIII shows the running-time of each step for testing a query (averaged over the 800 queries in the testing dataset) on a PC with 512M of memory and a 3.2Ghz CPU. We do not show the time for Step 4, since its running-time is negligable as compared to other steps. For example, when combining the results from any two classifiers, the time for each query is about 1.8×10^{-5} seconds.

From Table VIII, we can observe that the bottleneck of the test stage is to retrieve pages from search engines, which includes the time for finding and crawling the related pages taken by the search engine. For the search engine we developed based on Lemur and the crawled ODP pages (denoted by Lemur + ODP), the time for page retrieval is far less than that of Google and Looksmart,

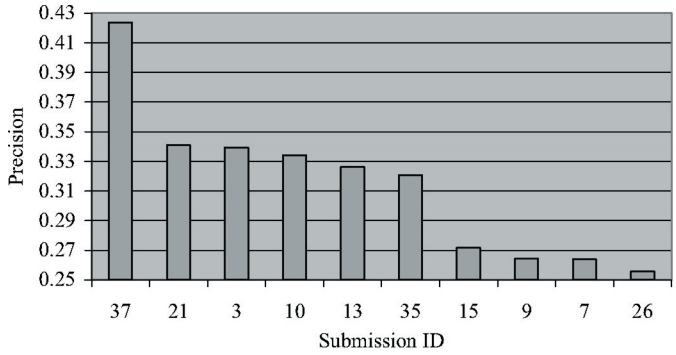
since the pages are indexed locally and we do not have to crawl them. In fact, the difference in time depends on the speed of the network. Therefore, when we run our approach on the sever of a search engine, a query can be classified in real time (in the order of 10^{-1}).

5.4.6 *Comparison with Other Participants.* To further validate our algorithm, we compare it with other KDDCUP2005 participants' systems. First, we briefly introduce two systems of the runner-ups in KDDCUP2005 competition [Kardkovács et al. 2005; Vogel et al. 2005] . The runner-up for the F1 criterion is the team from MEDai/AI Insight/Humboldt University. The runner-up for the precision criterion is from Budapest University of Technology.

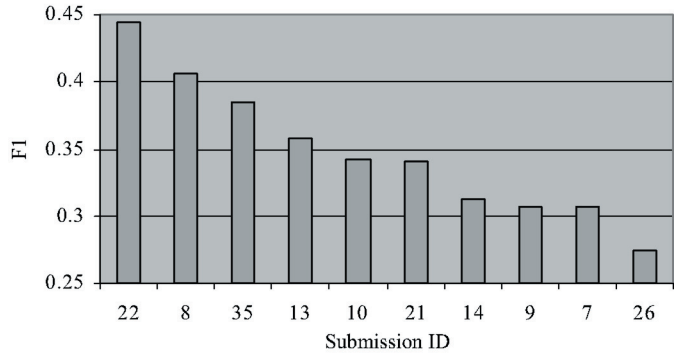
The MEDai/AI Insight/ Humboldt University team built a model containing 67 biclass classifiers, each of which corresponds to a target category. This model can predict 67 probabilities for each query and categories with the highest are chosen for each query. To build the model, they need to train 67 biclass classifiers. The training dataset for each classifier consists of the 111 queries given by the KDDCUP2005 organizers. For each of these queries, if it belongs to category i, it is a positive training sample of the classifier of category i; otherwise, it is negative. They send each query to the Google search engine and get some relevant pages. They use these retrieved pages to represent the query. With these training datasets, they train 67 biclass SVM classifiers. Given a test query, they input it to the classifiers, and then they can get the probability of each category. They choose the top categories as the final submitted result.

In the KDDCUP2005 competition, the Budapest University of Technology team proposed an algorithm called the Ferrety, which is based on Internet search engines and a document categorizer [Tikk et al. 2005]. As we observed, they also find that both the query words and KDDCUP2005 categories lack semantics, while no training data is available. Thus, they seek the meanings of words by asking the Internet, which acts as an open dictionary and knowledge base. They send each query to search engines and retrieve possible categories defined by them. Since the obtained categories differ from target categories, proper category mapping algorithms are needed. By observing that formal definitions of categories are available for many search engines, they perform mapping as follows: First, relevant word collections are expanded by WordNet synonyms of target category names; second, probable search engine categories are attached to target ones by matching the similarities in the word collections and category definitions. By calculating TF-IDF, more relevant words can be added to the collections. This process is repeated until the word collections become stable. Every search engine category is then mapped to a proper category defined by KDDCUP2005. Half of the 800,000 queries have results from search engines. The remainder are sent to their own document categorizer, which is a supervised algorithm for classifying text. They get another 320,000 answers with their categorizer. The precision of their algorithm is 0.34088.

Figure 14 contains the evaluation results for the top ten submitted solutions, as ranked by the organizers. For the F1 criterion, our value is 0.444 (submission ID: 22), that of the MEDai/AI Insight/Humboldt University team is 0.405 (submission ID: 8). Our F1 value is higher than that of the runner-up team by



(a) Precision of the top 10 solutions.



(b) F1 of the top 10 solutions.

Fig. 14. Top 10 solutions in KDDCUP2005 in terms of precision and F1.

9.6% and higher than the mean of the other nine teams among the top ten by 32%. For the precision criterion, our value is 0.424 (submission ID: 37), while that of the Budapest University of Technology team is 0.341 (submission ID: 21). Our precision value is higher than that of the runner-up team by 24.3% and higher than the mean of the other nine top competitors by 40.3%. Besides this, we also compared precision and F1 values of our solutions to the mean values of all other participants. Our precision and F1 are 98.5% and 73.5% higher, respectively, than the averaged precision and F1 of the other participants.

5.5 Some Failed Methods

In fact, we have tried several other approaches that did not work well. Here is an example. The main idea is to build a bridge between a given query and the 67 KDDCUP2005 categories by counting the number of pages related to both. We submitted a query to search engines and got its related pages. This set of pages is denoted by Pq. Similarly, we can get the related pages of a category by directly using the category name as a query. This set of pages is denoted by Pc. In practice, each Pq includes 100 pages for each query, and each Pc includes 10,000 pages. Then we can define the similarity between the target

query and category as $\|P_q \cap P_c\|$, where $\|.\|$ is the size of a set. For each query, we can return the most related categories according to the similarity. However, this method does not seem to work. One possible reason is that there is correlation between some pairs of categories. For example, we found that the overlap of the 10,000 retrieved pages of the categories "Computer\Hardware" and "Living\Tools and Hardware" consists of about 1000 pages. Therefore, we removed the overlap pages between each pair of categories. We repeated the aforementioned approach. However, the final result is still not satisfactory. One reason for the failure of this method is that we cannot judge the similarity between a query and a category simply by the size of the intersected set of retrieved pages. The essence of the problem is how to automatically find the exact collection of pages that can well represent the semantics of a query and a category, and how to determine the similarity between them based on these pages. This is a problem that requires further study.

We also tried another method, based on a dictionary, which also does not work well. We submitted a query q into dictionary software, for example, WordNet, to get the related words of a query. The result is denoted as R_q. The result includes the synonyms or antonyms of the given query, which can be a verb, noun, or adjective. Taking the query "car" as an example, the result contains: car, auto, automobile, machine, motorcar, railcar, railway car, railroad car, cable car, etc. Similarly, we can get the result of every category c in the same way, which is denoted as R_c. A similarity between the target query and category is defined as $\|R_q \cap R_c\|$, as shown before. We can classify a query into the top categories ranked according to similarity. When we tested this method on the validation dataset, the F1 is only about 20%. The main reason for the bad performance is that this method cannot obtain proper results for many queries through WordNet. If additional dictionaries such as Wikipedia are leveraged, the performance of this method may be improved.

6. CONCLUSION AND FUTURE WORK

In this article, we presented our approach to solving the query classification problem with its application on the task provided by KDDCUP2005. Query classification is an important as well as difficult problem in the field of information retrieval. Once the category information for a query is known, a search engine can be more effective and return more representative Web pages to users. However, since queries usually contain too few words, it is hard to determine their meanings. Another practical challenge, as shown in KDDCUP2005, is that no training data is explicitly provided for the classification task.

To solve the query classification problem, we designed an approach based on query enrichment that can map queries to some intermediate objects. With the intermediate objects, we designed several ensemble classifiers based on two different kinds of base classifiers, statistics-based and synonym-based. The experimental results on the two datasets provided by the KDDCUP2005 organizers validate the effectiveness of the base classifiers, as well as the ensemble strategies. We have designed a demonstration system called Q^2C@UST with a dedicated Web site at http://q2c.cs.ust.hk/q2c/.

The success of our approach in the KDDCUP2005 competition can be attributed to two factors: one is the way in which to enrich the queries and the other is the way to combine the base classifiers. Therefore, in the future, we will conduct more research following these two directions: (1) we will try to find additional valuable information for the queries on which we can build base classifiers, including those from some search engines with special features (such as that found at http://vivisimo.com/, which clusters the returned pages); and (2) we will conduct further research to find more effective strategies to generate ensemble classifiers.

APPENDIX A: Target Categories from the KDDCUP2005 Task Force

Computers\Hardware
Computers\Internet & Intranet
Computers\Mobile Computing
Computers\Multimedia
Computers\Networks & Telecommunication
Computers\Security
Computers\Software
Computers\Other
Entertainment\Celebrities
Entertainment\Games & Toys
Entertainment\Humor & Fun
Entertainment\Movies
Entertainment\Music
Entertainment\Pictures & Photos
Entertainment\Radio
Entertainment\TV
Entertainment\Other
Information\Arts & Humanities
Information\Companies & Industries
Information\Science & Technology
Information\Education
Information\Law & Politics
Information\Local & Regional
Information\References & Libraries
Information\Other
Living\Book & Magazine
Living\Car & Garage
Living\Career & Jobs
Living\Dating & Relationships
Living\Family & Kids
Living\Fashion & Apparel
Living\Finance & Investment
Living\Food & Cooking
Living\Furnishing & Houseware
Living\Gifts & Collectables
Living\Health & Fitness
Living\Landscaping & Gardening
Living\Pets & Animals
Living\Real Estate
Living\Religion & Belief
Living\Tools & Hardware
Living\Travel & Vacation
Living\Other
Online Community
Online Community\Chat & Instant Messaging
Online Community\Forums & Groups
Online Community\Homepages
Online Community\People Search
Online Community\Personal Services
Online Community\Other
Shopping\Auctions & Bids
Shopping\Stores & Products
Shopping\Buying Guides & Researching
Shopping\Lease & Rent
Shopping\Bargains & Discounts
Shopping\Other
Sports\American Football
Sports\Auto Racing
Sports\Baseball
Sports\Basketball
Sports\Hockey
Sports\News & Scores
Sports\Schedules & Tickets
Sports\Soccer
Sports\Tennis
Sports\Olympic Games
Sports\Outdoor Recreations
Sports\Other

ACKNOWLEDGMENTS

We thank Hong Kong University of Science and Technology, Department of Computer Science and Engineering and Computer Systems Group for their kind support.

REFERENCES

ALPAYDIN, E. 2004. *Introduction to Machine Learning*. MIT Press, Cambridge, MA.

BAUER, E. AND KOHAVI, R. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *J. Mach. Learn. 36*, 1–2, 105–139.

BEEFERMAN, D. AND BERGER, A. 2000. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York, 407–416.

BEITZEL, S. M., JENSEN, E. C., FRIEDER, O., GROSSMAN, D., LEWIS, D. D., CHOWDHURY, A., AND KOLCZ, A. 2005. Automatic web query classification using labeled and unlabeled training data. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, New York, 581–582.

CANN, A. J. 2003. *Maths from Scratch for Biologists*. John Wiley & Sons, New York, NY.

CARUANA, R., NICULESCU-MIZIL, A., CREW, G., AND KSIKES, A. 2004. Ensemble selection from libraries of models. In *ICML '04: Proceedings of the 21st International Conference on Machine Learning*. ACM Press, New York, 18.

CHANG, C.-H. AND HSU, C.-C. 1998. Integrating query expansion and conceptual relevance feedback for personalized web information retrieval. In *WWW7: Proceedings of the 7th International Conference on World Wide Web 7*. Elsevier Science Publishers B. V., Amsterdam, 621–623.

CHEKURI, C., GOLDWASSER, M., RAGHAVAN, P., AND UPFAL, E. 1997. Web search using automated classification. *6th International World Wide Web Conference (WWW6)*. Poster presentation.

CHEN, H. AND DUMAIS, S. 2000. Bringing order to the web: Automatically categorizing search results. In *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, 145–152.

DIETTERICH, T. G. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*. 1–15.

FAN, W., STOLFO, S. J., AND ZHANG, J. 1999. The application of adaboost for distributed, scalable and on-line learning. In *KDD '99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York, 362–366.

HANSEN, L. K. AND SALAMON, P. 1990. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell. 12*, 10, 993–1001.

HITEC. http://categorizer.tmit.bmr.hu.

HOEL, P. G. 1966. *Elementary Statistics*, 2nd ed. Wiley, New York, NY.

HOWE, A. E. AND DREILINGER, D. 1997. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Mag. 18*, 2, 19–25.

JANSEN, B. J. 2000. The effect of query complexity on web searching results. *Inf. Res. 6*, 1.

JOACHIMS, T. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *ECML'98 Proceedings of the 10th European Conference on Machine Learning*, 137–142.

JOACHIMS, T. 1999. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the 16th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, 200–209.

JONES, K. S. 1971. *Automatic Keyword Classifications for Information Retrieval*. Butterworth, London, UK.

KANG, I.-H. AND KIM, G. 2003. Query type classification for web document retrieval. In *SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM Press, New York, 64–71.

KARDKOVÁCS, Z. T., TIKK, D., AND BÁNSÁGHI, Z. 2005. The ferrety algorithm for the kdd cup 2005 problem. *SIGKDD Explor. Newsl. 7*, 2, 111–116.

KITTLER, J., HATEF, M., DUIN, R. P. W., AND MATAS, J. 1998. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 3, 226–239.

LEWIS, D. D. AND GALE, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR*, W. B. Croft and C. J. van Rijsbergen, Eds. Springer Verlag, Berlin, Germany, 3–12.

LI, Y., ZHENG, Z., AND DAI, H. K. 2005. Kdd cup-2005 report: Facing a great challenge. *SIGKDD Explor. Newsl. 7*, 2, 91–99.

MCCALLUM, A. AND NIGAM, K. 1998. A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*.

MEYER, D. A. AND BROWN, T. A. 1998. Statistical mechanics of voting. *Phys. Revei. Lett. 81*, 8, 1718–1721.

MILLER, G., BECKWITH, R., FELLBAUM, C., GROSS, D., AND MILLER, K. 1990. Introduction to wordnet: An on-line lexical database. *Int. J. Lexicography 3*, 4, 23–244.

PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. 1998. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford Digital Library Technologies Project.

SELBERG, E. AND ETZIONI, O. 1995. Multi-service search and comparison using the MetaCrawler. In *Proceedings of the 4th International World-Wide web Conference*. Darmstadt, Germany.

SHEN, D., PAN, R., SUN, J.-T., PAN, J. J., WU, K., YIN, J., AND YANG, Q. 2005. Q2c@ust: Our winning solution to query classification in kddcup 2005. *SIGKDD Explor. Newsl. 7*, 2, 100–110.

SHEN, D., SUN, J.-T., YANG, Q., AND CHEN, Z. 2006. Building bridges for web query classification. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

SILVERSTEIN, C., MARAIS, H., HENZINGER, M., AND MORICZ, M. 1999. Analysis of a very large web search engine query log. *SIGIR Forum 33*, 1, 6–12.

TIKK, D., BIRÓ, GY., AND YANG, J. D. 2005. Experiments with a hierarchial text categorization method on WIPO patent collections. In *Applied Research in Uncertainty Modelling and Analysis*, N. O. Attok-Okine and B. M. Ayyub, Eds. International Series in Intelligent Technologies, vol. 20, Springer-Verlag, 283–302.

VAN, R. C. 1979. *Information Retrieval*, 2nd ed. Butterworth, London, UK.

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. *SIGKDD Explor. Newsl. 7*, 2, 117–122.

VOORHEES, E. M. 1994. Query expansion using lexical-semantic relations. In *SIGIR '94: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer Verlag, Berlin, Germany. 61–69.

WEN, J.-R., NIE, J.-Y., AND ZHANG, H.-J. 2002. Query clustering using user logs. *ACM Trans. Inf. Syst. 20*, 1, 59–81.

YANG, Y. 1999. An evaluation of statistical approaches to text categorization. *Inf. Retr. 1*, 1–2, 69–90.

YANG, Y. AND PEDERSEN, J. O. 1997. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the 14th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 412–420.

Received November 2005; revised April 2006; accepted June 2006