Q^{2}C@UST: Our Winning Solution to Query Classification in KDDCUP 2005

Dou Shen, Rong Pan, Jian-Tao Sun,
Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin and Qiang Yang

Department of Computer Science, Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong, China

{dshen, panrong, jtsun, panjf, khwu,yinjie,qyang}@cs.ust.hk

http://webproject1.cs.ust.hk/q2c/

ABSTRACT

In this paper, we describe our ensemble-search based approach, *Q^{2}C@UST* (**http://webproject1.cs.ust.hk/q2c/**), for the query classification task for the KDDCUP 2005. There are two aspects to the key difficulties of this problem: one is that the meaning of the queries and the semantics of the predefined categories are hard to determine. The other is that there are no training data for this classification problem. We apply a two-phase framework to tackle the above difficulties. Phase I corresponds to the training phase of machine learning research and phase II corresponds to testing phase. In phase I, two kinds of classifiers are developed as the 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 are collected through the use of search engines. In the second stage, the enriched queries are classified through the base classifiers trained in phase I. Based on the classification results obtained by the base classifiers, two ensemble classifiers based on two different strategies are proposed. The experimental results on the validation dataset help confirm our conjectures on the performance of the Q2C@UST system. In addition, the evaluation results given by the KDDCUP 2005 organizer confirm the effectiveness of our proposed approaches. The best F1 value of our two solutions is 9.6% higher than the best of all other participants' solutions. The average F1 value of our two submitted solutions is 94.4% higher than the average F1 value from all other submitted solutions.

Keywords

Query classification, Synonym-based Classifier, ensemble learning, KDDCUP 2005.

1. INTRODUCTION

Historically, search engine technologies and automatic text classification techniques have progressed hand in hand. Ever since the early papers by the pioneers [14][18][22], people have recognized the possibility of conducting Web search through classification, and vice versa [2][5][6][15]. The KDDCUP 2005 competition made this connection even stronger; the task being to automatically classify 800,000 of the queries on a Web search engine into a set of 67 predetermined categories provided by the organizers. This task deviates from the traditional machine learning and text classification formulation in the following aspects:

- There are no training data available. It is part of the task to collect the training data. In fact, the manually annotated Web pages organized into topical directories could serve as the training data. However, different parts of the Web may provide training data of different quality. In contrast, text classification in machine learning and information retrieval has always included the training data as part of the input. How to make the best use of the Web, including its directory resources and search tools, provides a fresh perspective.
- The data are noisy and provide poor information. Most queries are short. There are some mis-spelt queries. Many words have multiple meanings and belong to several categories. For example, "office" can mean the working place as well as a kind of software. In contrast, several queries may in fact mean the same thing, such as "mainboard" and "planar board".
- The target categories suffer a shortage of semantic meanings. The 67 categories are provided without further explanation. Therefore, simple exact matches between queries and category names would be certain to fail.

Despite these difficulties, we are also encouraged by the possibility of answering some exciting questions through completing this challenging task. We have the following questions:

- How can machine learning, in particular classification, help in designing better search engines?
- Conversely, how can search engines help us design better text classifiers?
- How can the use of different training data impact on the quality of classification?
- How can similarity-based classification and statistics based classification be best integrated together?

In the sections that follow, we explain our choice of solutions as well as the results of testing these solutions on the validation and test data set provided by the competition organizers. One innovative aspect of our solution is the use of an ensemble of search engines to compose the classifiers, and an ensemble of classifiers to classify the massive input queries. The ensemble of search engines and the final integration of the search result showed that with moderate validation data, we can achieve a high level of performance when we appropriately combine the query categorization results.

2. OUR APPROACH

As in any machine learning application, we also adopt two phases in our solution. In phase I, which corresponds to the training phase of machine learning research, we collect data from the Web

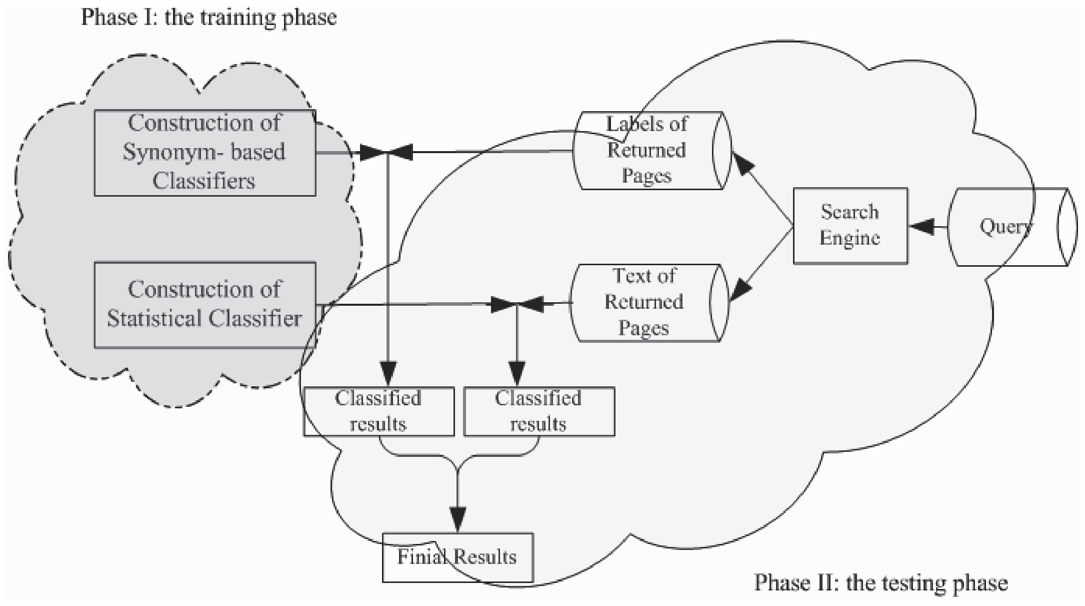

Phase I: the training phase
Construction of Synonym- based Classifiers
Construction of Statistical Classifier
Labels of Returned Pages
Text of Returned Pages
Search Engine
Query
Classified results
Classified results
Finial Results
Phase II: the testing phase

Figure 1. The architecture of our approach *Q^2C@UST*

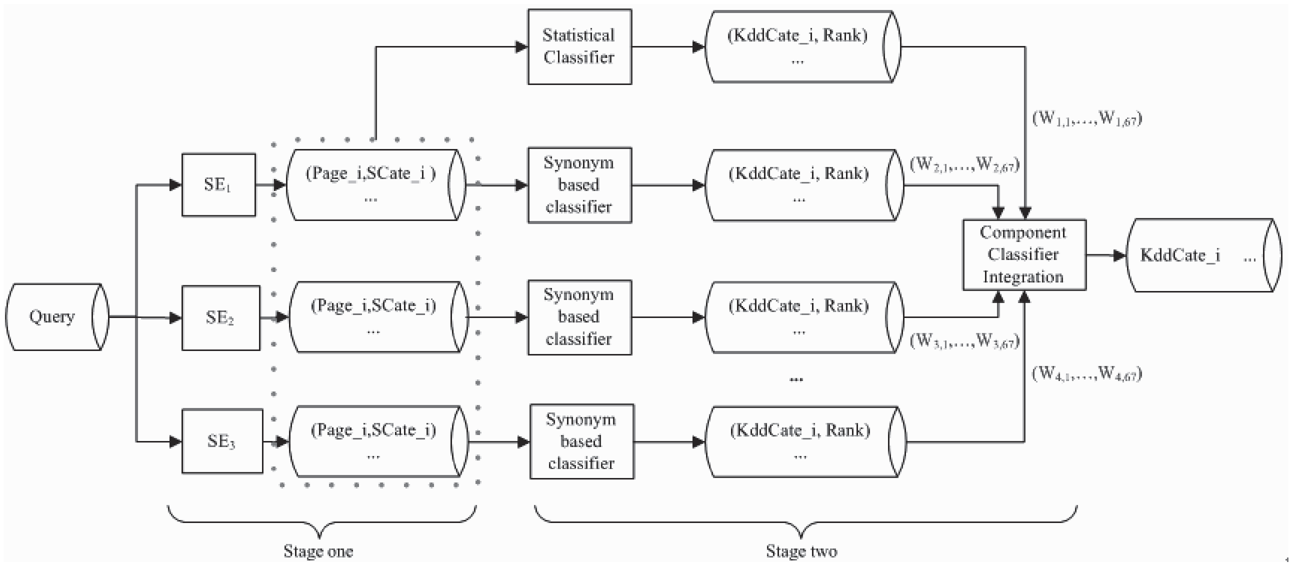


Figure 2. The two-stage framework in the "testing phase" of our approach *Q^2C@UST*

for training two base classifiers that map a query to the 67 categories provided by the KDDCUP 2005 organizers (KDDCUP categories).

In phase II, which corresponds to the "testing phase" in machine learning research, we process each query through a two-stage framework to handle the problem of query classifications. The overall framework of our approach is summarized in **Figure 1** and the detailed illustration of phase II is shown in Figure 2.

Phase II consists of two stages performed in a row. The first stage is to enrich the queries by searching the relevant pages which can provide the meanings for the queries in order for them to be classified. Enrichment is necessary because the queries are rather short, and thus their meanings ambiguous. We perform the enrichment process by finding the relevant text from related Web pages or the category information of these pages through Web search.

The second stage of phase II is to classify the queries based on the data collected in the first stage and using the classifiers trained in the first phase. At this stage, based on the classification results through the two kinds of base classifiers, two ensemble classifiers with different ensemble strategies are employed to classify the queries.

The experimental results show that the ensemble classifiers can improve the classification performance significantly.

2.1 Phase I: Training base Classifiers

We now discuss phase I of our approach: training classifiers for query classification. As noted above, a main problem of our task is the lack of training data. This is a new problem in machine learning: without training text documents with class labels, many previous methods cannot be used.

We tackle this problem by developing two kinds of base classifiers, which we later combine in an ensemble-search based classifier (Section 2.3). The first kind of base classifier is the synonym-based classifiers which uses the category information associated with Web pages collected for each query. The second kind of base classifier is statistical classifiers in which Support Vector Machine (SVM) is employed. We obtain two ensemble classifiers by combining these two kinds of classifiers according to different ensemble strategies, which improves the classification performance significantly more than the base classifiers.

2.1.1 *Synonym-based Classifiers*

As discussed in Section 2.1, after the query enrichment stage, we obtain a ranked category list for each query through a search engine. Since the category hierarchies from different search engines are distinct, the categories in different ranked lists vary a lot. In addition, the hierarchies of the search engines differ greatly from that given in the KDDCUP 2005 competition task. For convenience, we call the former *the search engine space* for a search engine, and the latter *the KDDCUP space*. For a particular search engine, our objective is to build a mapping function between the two spaces. Using this mapping function, we can classify a query into the 67 KDDCUP categories.

The mapping function can be built by keyword matching. We compare the keywords of categories in the KDDCUP space with those in the space of a certain search engine. Consider two categories, c_1 and c_2, where c_1 is from the space of KDDCUP and c_2 is from the space of a certain search engine. If they share some of the same keywords, we can map c_2 to c_1 directly. The categories in the space of KDDCUP have two levels, such as, "Computers\Hardware" and "Entertainment\Other". The second level specifies a particular field within the first level. For most of the categories in the space of KDDCUP, we only consider the keywords at the second level because they are not confused with other categories. A typical example is "Computers/Internet & Intranet". Even when we do not consider the first level for all the categories, there are no other categories which can be confused with "Internet & Intranet".

However, there are many categories that are more difficult to deal with. For them we require that the keywords in the first level and the second level should simultaneously match the categories in the space of a certain search engine. Otherwise, we cannot distinguish between two categories that share the same keywords only in the second level, such as "Computers/Hardware" and "Living/Tools & Hardware". Although they both have the keywords "Hardware" in the second level, they belong to two different domains "Computers" and "Living" in the first level. We give two examples to illustrate the above two cases:

- "…/internet/…" is mapped to "Computers/Internet and Intranet";
- "…/Computers/…./Hardware/…" is mapped to "Computers/Hardware" while "…/Living/…/Hardware/…" is mapped to "Living\Tools & Hardware".

In the implementation of the keywords matching method, we need to consider three special cases.

- On the second level of categories from the space of KDDCUP, the words connected by "&" can individually specify a concept. For example, in "Computers\Internet & Intranet", both "Internet" and "Intranet" can represent an individual concept. However, the words connected by a space can specify a concept only when they are joined together. One example is "Sports\Olympic Games" in which we need the two words together to define a concept. Therefore when matching the keywords, we take each word connected by "&" as a keyword and the combination of the words connected by space as keywords.
- We extend the keywords to include both the singular and plural forms in advance. For example "Living\Book & Magazine" is extended to "Living\Book & Magazine & Books & Magazines". Then we can conduct exact matching without missing any possible mapping.
- Seven out of the 67 KDDCUP categories just have a keyword "Other" on the second level, such as "Computers/Other". It is impossible to determine their semantics without taking other categories into consideration. In our approach, given a KDDCUP category, say "Computers/Other", we firstly collect all the categories containing the first level keyword, "Computers", in the space of a search engine. Then we remove the ones which can be mapped to other KDDCUP categories with the same first level as the target categories.

After applying the above direct mapping procedure, we may still miss a large number of mappings; many categories in the space of a search engine do not occur in the space of KDDCUP although they share words with the same meanings. In response, we expand the keywords in each label in the KDDCUP categories through WordNet [25]. For example, the keyword "Hardware" is extended to "Hardware & Devices & Equipments" and the keyword "Movies" is extended to "Movies & Films".

Using different search engines, we might have different category spaces. For each space, we can construct a mapping function between it and the space of KDDCUP categories. After obtaining these mapping functions, we can perform query classifications based on the category lists of each query collected in the data collection stage. For each category list of a target query, we map the categories to the space of KDDCUP according to the corresponding mapping function. We accumulate the number of times for each category in the space of KDDCUP being mapped onto. Then we can obtain the categories for the target query in the space of KDDCUP together with their occurrence count. By ranking the categories in terms of the occurrence count, we get a ranked list of KDDCUP categories into which the target query can be classified. Based on different category lists and their corresponding mapping functions, we gain different classification results. Here, we refer this kind of classifier as a synonym-based classifier.

Based on the direct mapping approach, the synonym-based classifiers tend to produce results with high precision but low recall. They produce high precision because the synonym-based classifiers are based on the manually annotated Web pages and can utilize the classification knowledge of editors. Therefore, once a mapping function is constructed, it is highly probable the classification result is correct. For example, we have shown that the categories such as "…/Computers/…./ Hardware/…" are mapped to "Computers/Hardware". Therefore, once a Web page associated with a query falls in the category "Computers/Hardware/Storage/Hard_Drives", we can assign "Computers/Hardware" to the query with high confidence. They produce low recall because it is hard to find all the mappings from the search-engine categories to the KDDCUP categories by keyword mapping. About 80,000 of the 354,000 categories in the category space of Google are not mapped onto the space of KDDCUP. Therefore, we cannot map all the categories in the category list for a query to the 67 KDDCUP 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 is described in the next section.

2.1.2 *Statistical Classifiers*

As discussed in the previous subsection, the synonym-based classifiers have a low-recall drawback. In order to address this problem, we proposed to use the statistical classifier to help classify queries. The statistical classifier classifies a query based on its semantic content. Thus even if the synonym-based classifier fails, a query can also be classified by a statistical classifier. In this paper, we use the Support Vector Machine (SVM) classifier because of its high generalization performance for document classification tasks and is easy to implement [12][13] .

When SVM is used for classification tasks (e.g., document classification), the first step is to train a model using a set of training examples. Then the model can be used to classify other examples. Apparently, it is not straightforward to apply SVM for a query classification task. There are at least two problems as we have discussed. Firstly, there is no training data available, and we only have the names of the 67 categories. Secondly, it is easy for a human being to capture the semantic meanings of a query and to identify its categories. However, for a statistical classifier, we must put forward a method to represent a query, i.e., to construct a query's features.

The above problems are both dealt with in our approach. For the first problem, we collect the training examples for the 67 categories by leveraging the mapping function constructed in the synonym-classifier learning step and the manually labeled Web page directory, such as ODP, etc. For each KDDCUP category, we can find a set of categories from the ODP directory by the mapping function. Thus Web pages contained in those collected ODP categories are used as the training documents for the target KDDCUP category. After the training examples are collected for each KDDCUP directory, the SVM training algorithm can be used to train the classifiers. For the second problem, we use the results returned by a search engine to help represent a query. As discussed in section 2.1, most queries contain only a few terms. Therefore, it is common that a query may be relevant to several topics. By leveraging the search results returned by a search engine, we try to capture the multiple topics relevant with the issued query. We keep the top N results (the parameter N are studied in section 3) and use the aggregate terms of the corresponding snippets, titles, URLs terms and the category names in the directory to represent the query. Finally, the query's bag of terms is processed by stop-word removal, stemming and feature selection. The resultant term vector can be used as input for the SVM classifiers and a ranked list of categories for each query is produced.

In our approach, the ODP directory is used for the purpose of collecting training documents. The SVM^{light} software package is used for training (http://svmlight.joachims.org/). The linear kernel is used and the one-against-rest approach is applied for the multi-class case. The DF and the information gain methods are used for feature selection. For KDDCUP categories which contain too many training examples, we use a sampling approach to decrease the scale of SVM training.

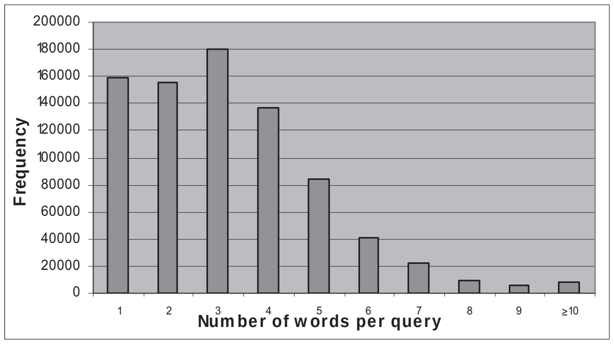


Figure 3. Frequency of queries with different length

2.2 Phase II, Stage I: Query Enrichment

In our second phase, we have two stages. In stage one, we take each query and enrich it so that there is a body of relevant text that represents this query. This step is a key task because our goal is to classify 800,000 queries into 67 categories *without any additional descriptions* about these queries.

The 800,000 queries vary a lot. They might be as simple as a single number such as "1939" or as complicated as a piece of programming code which involves more than 50 words. shows the statistical information about the number of words in each query and the frequencies in the 800000 queries. From this figure we can see that queries containing 3 words are most frequent (22%). Furthermore, 79% of queries have no more than 4 words. Each query is a combination of words, name of persons or locations, URLs, special acronyms, program segments and malicious codes. Some queries contain words which are very clean while others may contain typos or meaningless strings which is totally noisy. Some words may just have their meanings as defined in a static dictionary while others may have some special meanings when used on the Internet.

Moreover, the meaning of words may also evolve over time. For example, "apple" is a kind of fruit as given in a dictionary. However, it is also related to the name of a famous computer corporation nowadays. Thus, we cannot classify a query solely relying on a *static and out-of-date* training set. Instead, we should try to catch its meanings by asking the Internet, retrieving related documents and extracting its semantic features. What is more, in order 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 our approach, we send each query into Google [10], Looksmart [20] and a search engine developed by ourselves based on Lemur [17]. We chose these three search engines because they can provide both *directory search* and *Web search*. Directory search refers to search algorithms that return the related pages of a query together with the pages' categories. Since these categories of Web pages are labeled by people, it is appropriate to use them as the ground truth to classify the queries. However, not all the pages indexed by the search algorithm contain category information; in this aspect, Web search, which is used to refer to search algorithm as we typically use, can return more related pages that directory search cannot. Based on the content of the pages returned by Web search, we can classify the queries using a text classification algorithm. To enrich a query through search engines, we use the following three steps:

1. In step one, we first try to get the related pages through "Directory Search";
2. In step two, if we cannot get sufficient results from step one, we try to apply "Web Search" to the query to obtain the text;
3. If the retrieved results from Step two are still not enough, we check whether the queries are too noisy. If so, we use some preprocessing approaches to clean them up and repeat step 1 and step 2. Usually, after preprocessing on the noisy queries, we can obtain some meaningful and clean queries which make it possible to get new retrieved results through step 1 and step 2.

We use Google as an example to illustrate the three steps as shown in **Figure 4**.

- At the root level, all 800,000 queries are preprocessed by removing special characters such as "@,#,%" while keeping letters and digits. These 800,000 queries are first sent into the Google Directory Search. We can see from **Figure 4** that we are able to retrieve related pages for 500,000 (63%) of all queries in this way.
- We then send the remaining 300,000 queries into the Google Web Search and obtain results for the 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 the search for the most relevant queries to the original ones. For example, given the query "a cantamoeba", neither Google "Directory Search" nor "Web Search" returned any results. However, by trying the function "Did you

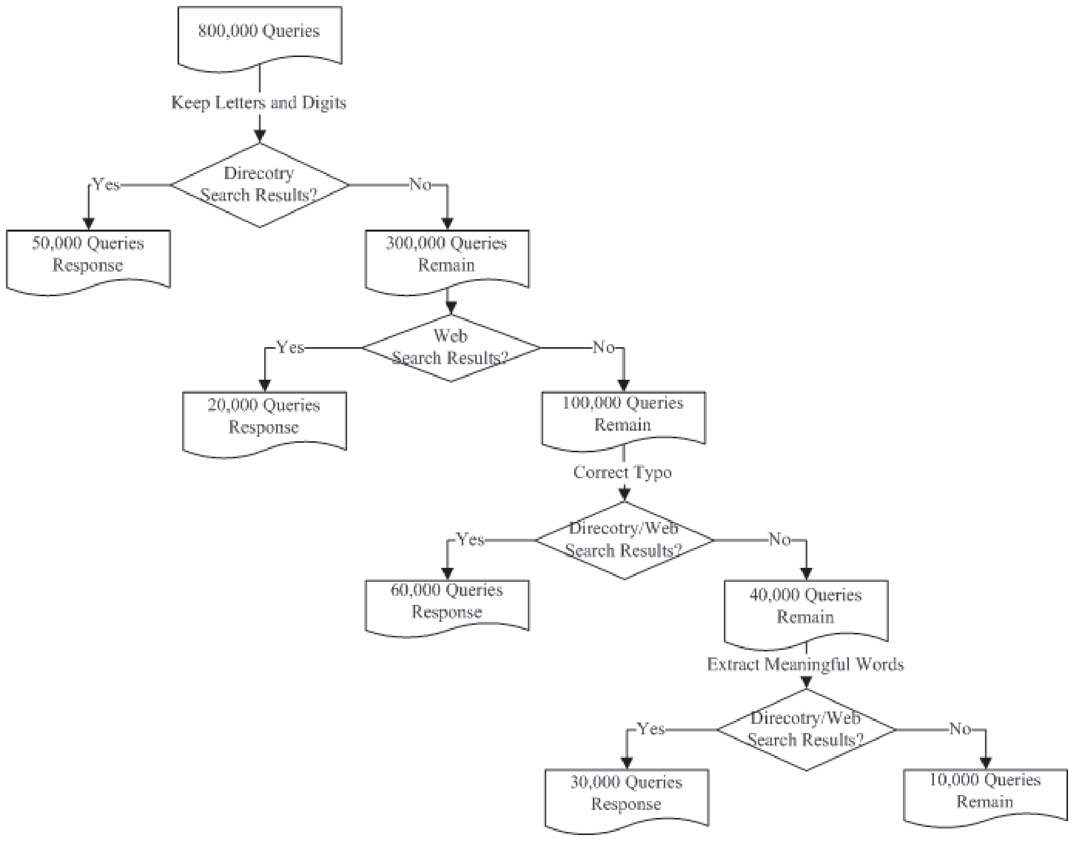


Figure 4. Illustration of the three steps for query enrichment

mean…", Google can return the 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 Google's "Directory Search"or "Web Search".

- However, after this step, there are still 40,000 queries left without any results. These queries are very noisy. They are usually connected words without spaces, long meaningless clobbers, or URL addresses containing parameters or even malicious codes. We try to convert theses queries into meaningful ones by extracting words from them through a maximum matching method based on the dictionary WordNet [25]. This method tries to extract as many meaningful words as possible and any extracted words should be as long as possible.

  Take the query "wheelbearingproblemsdamage" as an example, Google can not return any results through either "Directory Search", "Web Search" or even "Did you mean…". Therefore, we can split the whole query into four meaningful words "wheel bearing problems damage". After that, we can get reasonable results from Google "Directory Search" or "Web Search". In this way, we can get the results for 30,000 out of the remaining 40,000 noisy queries.

Based on the above procedures, there are only 10,000 queries without any pages returned from Google. These queries are inherently noisy and meaningless, such as "dddddfdfdfdfdf". We do not have to deal with these queries.

Note that in the query processing tree shown in **Figure 4**, the quality of the retrieved documents at different leave nodes may be different; the closer a node is to the root, the higher the quality of the result. Thus, the results of the 700,000 queries by applying "Directory Search" and "Web Search" are of high quality and reflect the true meanings of these queries. However, the results of the other 100,000 may not be reliable because we revise the queries to some extent.

We follow the same steps when using Looksmart [20]. Among the 800,000 queries, about 200,000 queries have "Directory Search" results. 400,000 have "Web Search" results and the remaining 200,000 have no results.

The third engine we use was developed by ourselves based on search engine constructor Lemur [17]. We first crawled 2 million Web pages with the category information from Open Directory Project (ODP) [21]. Then we indexed the collection of these pages with Lemur. Given a query, Lemur can retrieve a number of pages which are most relevant to the query together with their corresponding categories. Therefore the function of this search engine based on Lemur 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 all but 35.000 of the 800,000 queries..

In summary, after enriching the queries through a search engine, we can obtain two lists for representing each query. One is the relevant Web pages list. The other is a category list corresponding to the Web pages in the Web page list.

2.3 Phase II, Stage II: Query Classification through Ensemble Classifiers

From Phase I, we have constructed various classifiers which can classify the input queries into the KDDCUP categories. In this section, we discuss how we combine the results of these classifiers in the query-classification phase (testing phase) by an ensemble classifier, so that we can complete the task of query classification.

An ensemble is a set of models whose predictions are combined by weighted averaging or voting. Dietterich states that "A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse." [7] Recent work shows that models trained with different learning algorithms often make uncorrelated errors. Therefore, an ensemble of good models trained with different learning algorithms often outperforms the best model trained by one of the learning algorithms [4]. Moreover, many empirical investigations have shown that ensemble learning methods often lead to significant improvements across a wide range of learning problems [1][3][8][9][23].

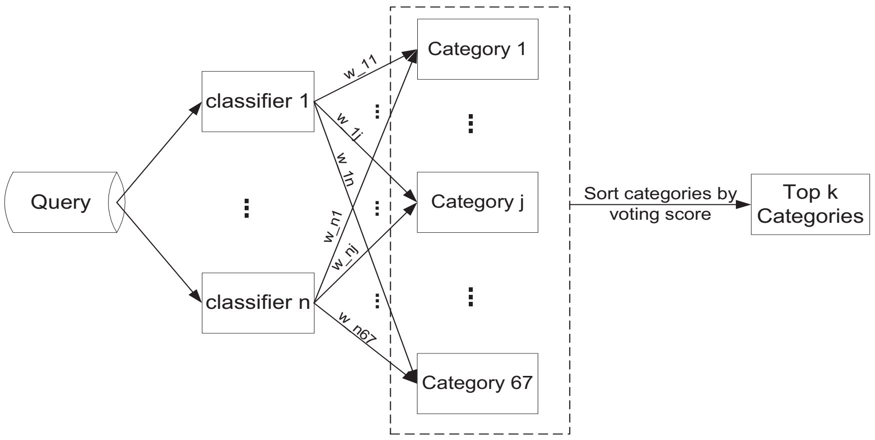


Figure 5. The ensemble paradigm

We use two approaches to training the ensemble classifiers. The first is to make use of the validation data set which contains 111 pairs of query-to-category mappings provided by the KDDCUP organizers, to adjust the combination weights. The second approach is to ignore this validation data set in order to avoid overfitting.

Our primary method in combining different classifier functions is to assign an ensemble weight value to the result returned by each classifier function, and rank the final list of classified categories. (See **Figure 5**) It is tricky to assign the ensemble weights. Different classifiers introduced in the above sections have different 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 the other category. 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 classes according to its performance. To determine the weights, we validate each component classifier on the 111 samples given by the KDDCUP organizers. The higher precision a classifier achieves on a category, the higher weight assigned to the classifier on that category. As a result, each classifier may obtain a weight value W_{ij} on a KDDCUP category j.

However, the 111 samples may be too small to be a suitable validation data set as it is easy to be overfitting on these samples. Therefore, a conservative way to combine the classifiers is to assign equal weights to them.

Both the weight-assignment strategies in ensemble classifier construction are compared in the following experiments.

3. EXPERIMENT AND DISCUSSION

To test our proposed approach, we conduct some experiments on the datasets provided by the KDDCUP 2005 organizer. The experimental results validate the effectiveness of our approach. In addition, we give some analysis of the consistency of the three labelers on their judgment of the performance of the classifiers in this part.

3.1 Dataset

The KCCCUP2005 organizer provides a dataset which contains 111 sample queries together with the manual categorization information. These samples help exemplify the format of the queries and provide the semantics of a tiny number of queries. In fact, since the category information of these queries is truthful, they can serve as the ground truth for the test data and validation data for our proposed classifiers. Later, after the competition is finished, to test the solutions submitted by the participants, the organizer provides another dataset which contains 800 queries with labels from three human labelers. We denote the three labelers (and sometimes the dataset labeled by them if no confusion is caused) as L1, L2 and L3, respectively. We refer to the former as Sample Dataset and the latter as Testing Dataset in the following sections.

3.2 Evaluation Criterion

The evaluation criteria adopted by the KDDCUP organizer is the standard measures to evaluate the performance of classification in Information Retrieval (IR), including precision, recall and F1-measure [24]. The definitions of precision, recall and F1 in the query classification context are given below:

A: $\sum_i \# \, of \;\; queries \; are \; correctly \; tagged \; as \; c_i$

B: $\sum_i \# \, of \;\; queries \; are \; tagged \; as \; c_i$

C: $\sum_i \# \, of \;\; queries \; whose \; category \; is \; c_i$

$$\mathrm{Pr}ecision = \frac{A}{B}$$

$$\mathrm{Re}call = \frac{A}{C}$$

$$F1 = \frac{2 \times \mathrm{Pr}ecision \times \mathrm{Re}call}{\mathrm{Pr}esion + \mathrm{Re}call}$$

For the Sample Dataset, we report the precision, recall and F1 evaluation results for each classifier. For the Testing Dataset, three labelers are asked to label the queries, thus the results reported on this dataset are the average values[19]. Take the calculation of F1 as an example:

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

3.3 Experimental Results and Explanation

As introduced in the previous section, we have a total of six classifiers, three synonym-based classifiers, one statistical classifier that is SVM and two ensemble classifiers. For simplicity, we refer to the three synonym-based classifiers which rely on Google, Looksmart, and Lemur as S1, S2 and S3 respectively. We denote the ensemble classifier which relies on the validation dataset as EV and the one that does not rely on validation data as VN. In the following part, we first investigate the two main parameters which affect the performance of our proposed classifiers on the sample dataset and then we compare the performance of those classifiers. Finally, we give some analysis of the consistency of the three labelers on their judgment of the classifiers' performance.

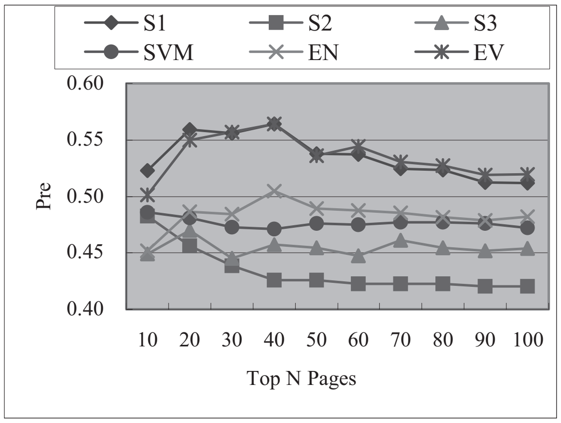
3.3.1 Parameter Tuning

There are two main parameters which significantly impact on the performance of our proposed classifiers. One is how many related pages we should use for each query. If we use too few pages, we may fail to cover all the diverse topics of a query. However if we use too many pages, we may introduce much noise. Another parameter is how many labels we should assign to each query. The competition rules allow us to assign at most 5 labels for each query. However, in order to achieve higher precision, we should assign fewer but very accurate labels. If we hope to achieve higher recall, we need to assign more possible labels.

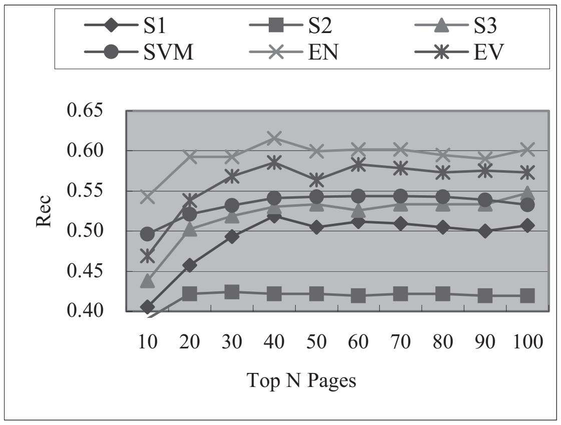
Figure 6 shows the performance of different classifiers with respect to an increasing number of related pages used for classification. Here, the related pages are taken into consideration in the order they are returned by the search engines, that is in the order of the degree of relevance with the query. The results shown in **Figure 6** verify our conjecture. As the number of related pages increases, the precision tends to decrease, although the precision for some classifiers increase at first. The reason is that we need a certain number of pages to get the meaning of the query. However, if we include too many pages, noise may be introduced which can reduce the precision. From **Figure 6**, we can see that the critical point for most classifiers is 40. Before that point, the performance of the classifiers increases when we use more and more pages, while after that point the performance begins to decrease. Therefore in the following experiments we keep only the top 40 pages for subsequent query classification.

Figure 7 shows the performance of different classifiers by varying the number of classified categories. As we expected, when the number of classified categories increases, the precision of all the classifiers decreases while the recall increases significantly. In contrast, the value of F1 increases at the beginning and then subsequently decreases. For most of the classifiers, the maximum values of F1 are achieved when four categories are generated for each query. Although the F1 values are close to those obtained when 5 categories are assigned, the precision values are much lower.

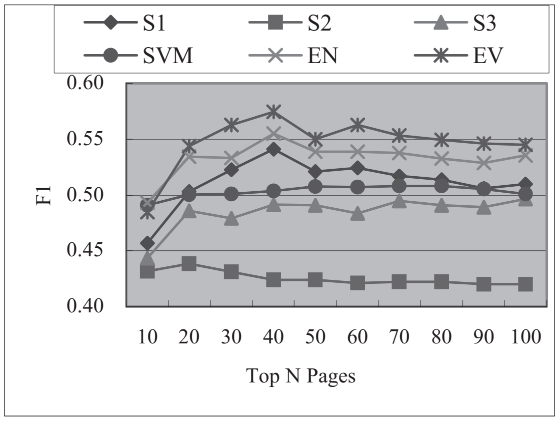
Based on the observation of the impact of two parameters, in order to achieve higher precision without sacrificing F1 much, we consider only the top 40 related pages of a query and return 4 categories for each query in most cases.



(1) Precision of the Six Classifiers

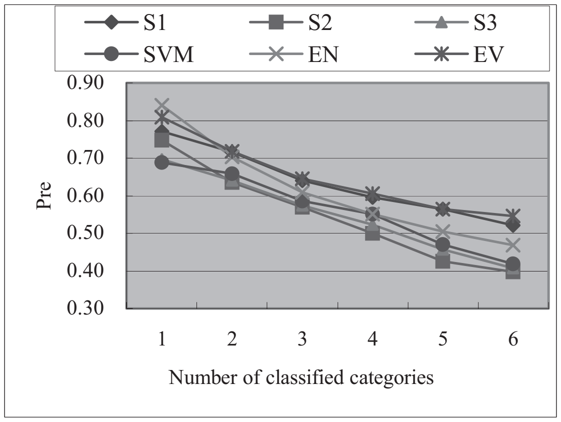


(2) Recall of the Six Classifiers

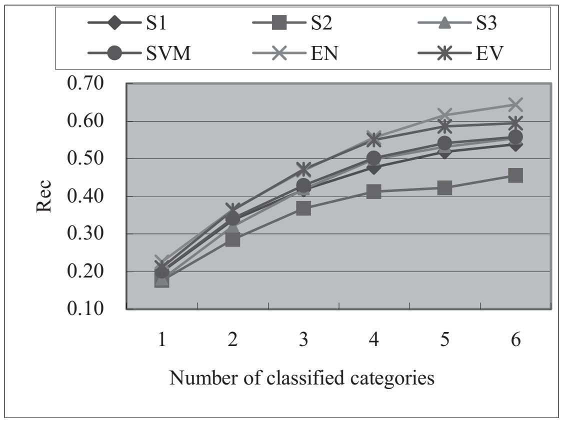


(3) F1 of the Six Classifiers

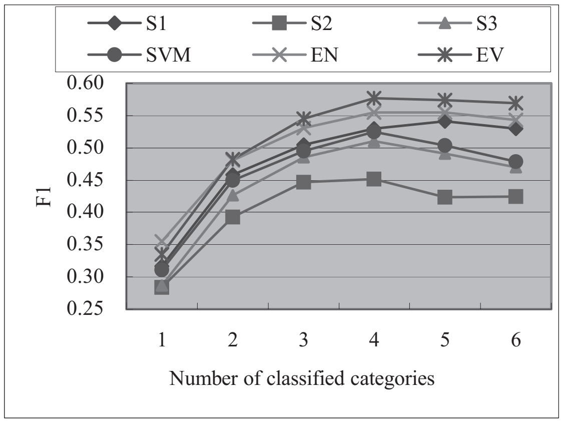
Figure 6. Performance of different classifiers with the number of used related pages on the 111-sample dataset



(1) Precision of the Six Classifiers



(2) Recall of the Six Classifiers



(3) F1 of the Six Classifiers

Figure 7. Performance of different classifiers with the number of classified categories on the 111-sample dataset

3.3.2 Comparison between Classifiers

From the above experimental results on the Sample Dataset, we can see that among the four base classifiers, S1 works best while S2 works worst. The reason why S2 does not work well is that for many queries, we cannot obtain enough related pages through the Looksmart search engine. For SVM, we expect that it can solve the low-recall problem caused by the three synonym-based classifiers, as discussed in section 2.1.1. **Figure 6** and **Figure 7** show that SVM obtains the highest recall in most cases. 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 (EN, EV and S1), we can see that, compared with S1, EN and EV improve the F1 by 4.53% and 8.67% respectively. In fact, when we design these two ensemble classifiers, EV is expected to achieve higher precision measure because each component classifier is highly weighted on the classes where 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 4.2% improvement compared with the F1 value of EV (0.426). While the precision value of EV (0.424) improves by 2.3% compared with that of EN (0.414).

The effectiveness of our proposed ensemble classifiers can also be validated when compared with the other participants' solutions on the Testing Dataset. The F1 value of our solution generated by EN is 9.6% higher than the best one among other participants' results. The averaged F1 value of our submitted results is 94.4% higher than the averaged F1 value of all the others.

3.3.3 Analysis of the Consistency between the Three Labelers

In this section, we analyze and discuss the consistency between the three labelers on their judgment of the performance of the classifiers.

Figure 8 shows the F1 values of the six classifiers developed by us on the testing data labeled by the three labelers. From Figure 8, we can see that the three labelers have a high correlation with respect to the relative performance of the classifiers, especially labeler 1 and labeler 3. After ranking the six classifiers according to each labeler, we calculate the spearman rank-order correlation coefficient between each pair of the labelers [11]. The Spearman correlation is a nonparametric approach to calculating the relationship between two variables based on ranking the two variables and no assumption about the distribution of the values is made. The results are shown in Table 1.

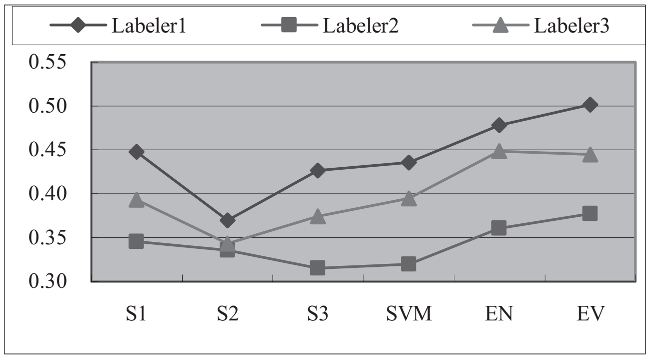


Figure 8. The distribution of the labels assigned by the three labelers

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

Table 1. Spearman correlation between each pair of labelers

In general terms, correlation coefficients over 0.67 indicate strong relationships. So we can conclude that the three labelers are in strong correlation when they determine the performance of classifiers.

3.4 The Failed Methods

In fact, we have tried several other approaches which do not work well. Here is an example. The main idea is to build a bridge between a query and the 67 KDDCUP categories by counting the number of pages relating to both of them. We submitted a query into search engines and got its related pages. This set of pages is denoted by P_q. Similarly, we can get the related pages of a category, using the category name directly as a query. This set of pages is denoted by P_c. In practice, each P_q includes 100 pages for each query, and each P_c includes 10,000 pages. Then we can define a 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 by similarity. However, this method does not seem to work. One possible reason is that there is a 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 & Hardware" are about 1000 pages. Therefore we removed the overlapping pages between each pair of categories. We repeated the above approach. However, the final result is not satisfactory. One reason for the failure of this method is that the retrieved pages cannot effectively represent the semantics of a category. 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. This is a problem that needs further study.

We also tried another method based on a dictionary which does not work well either. We submitted a query *q* into dictionary software, e.g., 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, a noun, or an adjective, among others. Take the query "car" as an example, the result contains: car, auto, automobile, machine, motorcar, railcar, railway car, railroad car and cable car, and so on. Similarly, we can get the result of every category *c* through 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 build a matrix and get the most relevant categories based on similarity for every query. When we test this method on the validation dataset, the F_1 is only about 20%. The main reason for the poor performance is that this method cannot obtain proper results for many queries through WordNet. If more dictionaries like Wikipedia are leveraged, the performance of this kind of method might be improved.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented our approach to solving the query classification problem provided by KDDCUP 2005. Query classification is an important as well as a difficult problem in the field of information retrieval. Once the category information for a query is known, the search engine can be more effective and can return more representative Web pages to the users. However, since the queries usually contain too few words, it is hard to determine their meanings. Another challenge in KDDCUP 2005 is that, no training data are provided for the classification task. What is more, the semantics of the predefined category structure in KDDCUP 2005 are not given explicitly.

To solve the task of KDDCUP 2005, we rely on some extra (external) information to overcome these difficulties. Therefore, we proposed an ensemble search based method which consists two phases with the second phase containing two stages. Phase I corresponds to the training phase of machine learning research and phase II corresponds to testing phase. In phase I, two kinds of base classifiers are developed. One is synonym-based and the other is statistics based. At the first stage of phase II, the queries are enriched in that the related Web pages together with their category information are collected for each query through search engines. In the second stage, the queries are classified through the classifiers trained in phase I. The synonym-based classifiers perform classification using the category information of the related Web pages for each query while the statistical classifier uses the contents of the related pages for each query. The experimental results on the two datasets provided by the KDDCUP 2005 organizer validate the effectiveness of these classifiers. By combining these two kinds of classifiers through two different strategies, we obtained two ensemble classifiers. One is expected to achieve higher precision with the help of the validation dataset and the other is expected to achieve a higher F1 without using the validation dataset. Both our experimental results on the datasets and the evaluation results given by the organizers verified the effectiveness of our approach.

Our proposed approach is proved to be very effective for the query classification problem. We have designed a demonstration system called *Q^2C@UST*, with a dedicated Web site at http://webproject1.cs.ust.hk/q2c/. Our success is due to two factors: one is the method for enriching queries and the other is the method of combining the base classifiers. In the future, we will conduct more research work following the two directions: 1) we would try to find more valuable extra information for the queries based on which we can build the base classifiers; 2)we will conduct some further research to find some more effective strategies to generate ensemble classifiers.

5. ACKNOWLEDGMENTS

We thank Hong Kong RGC Grant HKUST # 03/04.EG01 HKBU 2/03C, HKUST 6187/04E and Hong Kong University of Science and Technology, Department of Computer Science and Computer Systems Group for their kind support. We also thank KDDCUP 2005 organizers for their suggestions on this paper.

6. REFERENCES

[1] E. Bauer, R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. Machine Learning, 36:1/2, 105-142. 1999.

[2] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 407-415, 2000.

[3] L. Breiman. Bagging predictors. Machine Learning, 24:2, 123-140. 1996.

[4] R. Caruana and A. Niculescu-Mizil. Ensemble selection from libraries of models. In Proc. 21th International Conference on Machine Learning (ICML'04), 2004.

[5] C. Chekuri, M. Goldwasser, P. Raghavan and E. Upfal. Web Search Using Automated Classification. Poster at the Sixth International World Wide Web Conference (WWW6), 1997.

[6] H. Chen, S. Dumais. Bringing order to the Web: Automatically categorizing search results. In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), pages 145-152, The Hague, The Netherlands, April 2000.

[7] T. G. Dietterich. Ensemble methods in machine learning. First International Workshop on Multiple Classifier Systems, pages 1-15, 2000.

[8] W. Fan, S. Stolfo, J. Zhang. The application of AdaBoost for distributed, scalable and on-line learning. In Proceedings of the Fifth SIGKDD International Conference on Knowledge Discovery and Data Mining, 362-366. 1999.

[9] Y. Freund, R. E. Schapire. Experiments with a new boosting algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning, 148-156. 1996.

[10] Google, http://www.google.com

[11] P. G. Hoel, Elementary Statistics, Wiley, 1971.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In Proc. 16th International Conference on Machine Learning (ICML), Bled, Slovenia, June 1999.

[13] T. Joachims (1998): Text Categorization with Support Vector Machines: Learning with Many Relevant Features. European Conference on Machine Learning (ECML), Claire Nédellec and Céline Rouveirol (ed.), 1998.

[14] K. S. Jones. Automatic Keyword Classification for Information Retrieval. Butterworths, London, 1971.

[15] I.H. Kang, G. Kim, Query type classification for web document retrieval. In Proceedings of the 26rd annual international ACM SIGIR Conference on Research and Development in Information Retrieval. Toronto, Canada, 2003, 64-71.

[16] J. Kittler, M. Hatef, R. P.W. Duin, and J. Matas. On Combining Classifiers. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, 1998, pp. 226-239.

[17] Lemur, http://www.lemurproject.org/

[18] D.D. Lewis, W. A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, pages 3-12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[19] Y.Li, Z.J.Zheng, K.Dai. KDD-CUP 2005. Presentation on The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago, USA. August 21, 2005. http://kdd05.lac.uic.edu/kddcup.html.

[20] Looksmart, http://www.looksmart.com

[21] ODP: Open Directory Project, http://dmoz.com

[22] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, USA, 1998.

[23] J. R. Quinlan. Bagging, boosting and C4.5. In proceedings of the Thirteenth National Conference on Artificial Intelligence, 725-730. 1996.

[24] C.J. van Rijsbergen. Information Retrieval. Second Edition, Butterworths, London, 1979, 173-176.

[25] Wordnet, http://wordnet.princeton.edu/

About the authors:

Dou Shen is a Ph. D. student in the Department of Computer Science at Hong Kong University of Science and Technology. His research interest includes machine learning and data mining, especially in the field of sequential data mining and text mining.

Jian-Tao Sun is a Ph. D. candidate at the Department of Computer Science and Technology, Tsinghua University at Beijing, China. He was a visiting scholar at Hong Kong University of Science and Technology when he took part in the KDDCUP 2005 competition. His research interests include text mining, Web usage mining and kernel methods.

Rong Pan is a postdoctoral fellow at Hong Kong University of Science and Technology. His research interest includes machine learning, data mining and case-based reasoning.

Jeffrey Junfeng Pan is a Ph. D student in the Department of Computer Science at Hong Kong University of Science and Technology. His research interest includes machine learning and data mining. Currently, he is working on location estimation and activity recognition in wireless (sensor) networks and pervasive environments.

Kangheng Wu is a Ph. D. student in the Department of Computer Science at Hong Kong University of Science and Technology. His research interest includes automated planning and machine learning, especially in the field of learning action models from observed plans.

Jie Yin is currently a Ph. D. student in the Department of Computer Science at Hong Kong University of Science and Technology. Her research interests include artificial intelligence, pervasive computing and data mining. Currently she is working on location estimation and human behavior recognition from sensory data in pervasive environments.

Qiang Yang (http://www.cs.ust.hk/~qyang) is a faculty member in the Department of Computer Science at Hong Kong University of Science and Technology. His research interests are artificial intelligence, Web search, data mining and pervasive computing.