Kasetsart University TREC-10 Experiments

P. Norasetsathaporn and A. Rungsawang
{g4365020,fenganr}@ku.ac.th
***M**assive **I**nformation & **K**nowledge **E**ngineering*
Department of Computer Engineering
Faculty of Engineering
Kasetsart University, Bangkok, Thailand.

Abstract

We use WORMS, our experimental text retrieval engine, in our TREC-10 experiments this year. The Okapi weighting scheme is used to index and test in both web ad hoc and homepage finding tasks. In web ad hoc task, we propose a novel approach to improve the retrieval performance using text categorization. Our approach is based on an assumption that "Most relevant documents are in the same categories as their query". This retrieval approach has a great effectiveness in reducing the number of searching documents and the searching time. In addition, it can be used to improve precision of retrieval. In homepage finding task, we use a query expansion based method and the Google search engine to find more useful words to be added to the topics. Before using with homepage finding task, we test the query expansion method with the homepage finding training set to get the best type of expanded query.

1 Introduction

In our TREC-9 experiments last year [11], we modified the Cornell's SMART version 11.0, in addition with the notable pivoted unique normalization weighting scheme [3], to run smoothly on our Linux machine. However, we confronted with intrinsic operating system problem. We could not index the whole web track collection in one-shot. We therefore decided to split the web track collection into sub-collections. We indexed and tested all sub-collections separately and merged all subsequence results to get the final top-1000 scores to send to NIST.

In the TREC-10 experiments this year, we use WORMS [15], our own text retrieval engine, to index and experiment instead of using the Cornell's SMART version 11.0. WORMS can eliminate the intrinsic Linux operating system problem because it does not create inverted file image that is larger than 2 GB, as it can split the inverted file image into several smaller ones. We also implement the notable Okapi weighting scheme [13, 14] in WORMS and use it to index and test both web ad hoc and homepage finding tasks. In the web ad hoc task, we propose a novel approach to improve the retrieval performance using text categorization. We formulate an assumption and set the experiments to test that assumption. We study the relation between the relevant documents for a query and its categories using TREC10 as the test collection and Google Web Directory as the knowledge base for the SVM classifier [5, 8, 12]. This retrieval approach provides a great effectiveness in reducing the number of searching documents and the searching time. In addition, it can be used to improve precision of retrieval by re-ranking method. In homepage finding task, we use a query expansion technique to add more useful words to topics. Before we use it with homepage finding task, we tested the method with homepage finding training set to get the best type of expanded query.

This paper is organized in following way. Section 2 introduces shortly our information retrieval engine. Section 3 gives more detail what we do in web ad hoc task and about the experimental results. Section 4 describes the query expansion method we use in homepage finding task and the experimental results we obtain. Section 5 concludes this paper.

2 Information Retrieval Engine

An experimental information retrieval engine, WORMS [15] is a high-performance text retrieval system implemented using the PVM message-passing library, and running on the cluster of low-cost PC based machines. WORMS is based on the vector space indexing model and the inverted file structure for effective and fast retrieval. There are actually 3 prototypes of WORMS. The first one is the sequential WORM prototype that we use in this TREC10 experiments. The two others are the parallel WORMS and the high performance WORMS that will not be mentioned here. The 4 main components of the sequential WORM are Stop&Stem, Indexer, Invfile, and Retriever, respectively. WORM's indexer can read user's input parameter and create several chunks of document vector images and its corresponding inverted files, of which each file's size is less than 2 GB barriers of the Linux x86 operating system.

Okapi Weighting Scheme

We found during our TREC10 experiments that the pivoted normalized weighting scheme [3] we used in TREC-9 experiments last year are not as good as we expect. In case of the TREC-9 data, weighting it with Okapi's byte length normalization gives better retrieval effectiveness. Therefore, we use the Okapi's byte length normalization which is base on the 2-Possion distribution model [13, 14] in TREC-10 this year. The following weighting scheme is implemented in WORMS.

The o weight implemented in WORMS:

$$L(n,i) = \frac{f_{ni}}{2 \times (0.25 + 0.75 \times \frac{dl}{avg\_dl}) + f_{ni}} \tag{1}$$

The p weight implemented in WORMS:

$$\log \frac{N - df_i}{df_i} \tag{2}$$

where dl is the document length, avg_dl is the average document length, f_{ni} is the raw tf-factor, N is the total number of documents in the collection and df_i is the number of documents which a term i appears.

3 Web Ad Hoc Task

We perform 2 experiments for this task. First, we index and test the small web track collection (WT10g) using WORM and Okapi weighting scheme. Second, we use text categorization technique to increase the retrieval effectiveness. The rest of this section gives more detail of our propose text categorization technique.

3.1 Using Text Categorization

Owing to the incessant growth of the Internet and the abundant availability of text data in the World Wide Web, text classification is acquiring more popularity with the growing interest. Many search engines, such as Google[1], Yahoo[6] and AltaVista[7], provide data that has been grouped into categories or directories. In addition to managing data, text categorization is a great benefit in our research. We use text categorization to improve retrieval performance of web ad hoc task. Our method bases on the assumption that "Most relevant documents are in the same categories as their query". To evaluating our assumption, we have to study the relation between the relevant documents for a query and its categories.

The problem of this experiment is how we classify WT10g documents and queries into hierarchical structure. To solve this problem, the machine learning technique called "Support Vector Machine" (SVM) [5, 12] is used to automatically construct classifier by using the existing web directory of the Google, designed for human Web browsing, as the source of knowledge base for training and testing process. After training step, we obtain the top and the second level classifiers, one classifier for one category, trained by the Google Web documents. Following that classifiers, we classify WT10g collection and the topics number 501-550 into categories by following the Google Web Directory structure. Then we study the relation between the relevant documents for a topic and its corresponding category to test our assumption and use it to improve retrieval effectiveness.

Support Vector Machine

Support Vector Machines (SVM) [5, 12] is a relatively new learning approach, introduced by Vapnik in 1995, for solving the two-class pattern recognition problems. It is based on the Structural Risk Minimization principle for which error-bound analysis has been theoretically motivated. The method is defined over the vector space. The decision surface separates the data points into two classes. In this research, we employ the Joachim's implementation, SVM^{light} [8, 9], owing to its accuracy and effectiveness.

Google Web Directory

Google Web directory integrates search technology with Open Directory pages to create the most useful tool for finding information on the Internet. The Open Directory Project[2] is claimed to be the largest, most comprehensive human-edited directory of the Web database. It is constructed and maintained by a vast, global community of volunteer editors. The Google directory contains over 1.5 millions URLs. These URLs are organized in Google Web documents which connected together. The more general the category, the closer to the root of the tree it is. The connection of Google documents are quite complex. There are some category names linked to other Google categories, classified under a different path of the Google hierarchy.

Pre-processing Phase

For TREC10, we parse all html tags, images, all messy data and the others, out of the WT10g collection. We also remove stop-word and stem the rest [4] from the WT10g collection. In case of web ad hoc topics, we also remove stop-word and stem the rest to build queries. After that, WT10g documents are weighted with *nno* weighting scheme and queries are weighted with *npn* weighting scheme.

For Google collection, we create the local copy of Google Web directory by using Wget [1], a GNU network utility for retrieving files from the WWW. The document is stored in a hierarchy structure by Wget itself. From documents which contains hyperlink information between them, we build the same hierarchical structure as the Google Web directory. We have 2 methods for managing documents into categories. First, we choose only documents that are not cross-linked to other categories. Second, we choose all documents under the category, including those in their cross-links. In this paper, we choose the first method because most of the documents in Google hierarchy belong to many cross-linked categories. For this reason, each category content has not been obviously separated. When we classify queries into categories, most of the queries belong to many categories as well. There are 15 top-level and 431 second-level categories we choose to experiment. We do not include World top-level categories because most web pages in this category is not written in English. Then we remove all HTML tags from documents and extracted "entry title", the title of a categorized entry indexed in a category, and "entry summary", the brief textual description of an entry [10]. We also reduce the number of features by removing words contained in the stop word list and stemming the rest [4].

The feature selection methods we used to reduce a high dimensional feature space in Google Web Directory is Document frequency thresholding (DF) [17]. Because of its simplicity, effectiveness, and low cost in computation, DF was choosen. Document frequency is the number of documents in which a term occurs. We compute the document frequency for each unique term in the Google documents and remove the term with DF below a threshold (DF = 5) from the feature space.

Building Classifier

We build a classifier for each category in both levels. In each category, a training set and a testing set are chosen by applying the systematic random selection to the documents. For the top-level category, we select the same number of documents in each of the top sub-categories by choosing every n documents, where n calculated from dividing the number of documents in each sub-category by the number of documents we want. With this method, we obtain sample documents throughout sub-categories in the hierarchy. In the second-level, we use the same method but the training and testing examples only come from documents in the same top-level category. These sets are positive examples of the category. The negative examples are selected from the other categories. After the selection process, we obtain training and testing document sets for every category. The number of training and testing Google Web documents in the top-level and the second-level are shown in Table 1 and Table 2, respectively.

Then, Google training data were weighted by *nno* weighting schemes and Google testing data were weighted by *npn* weighting schemes. Finally, we create top-level classifiers and second-level classifiers from training sets we provided.

[1] http://www.lns.cornell.edu/public/COMP/info/wget

Category Name	**Total**	**Training**		**Testing**
		P	N	P+N
Arts	244232	39932	77436	98912
Business	194269	57052	66765	89351
Computers	104371	33454	76155	91117
Games	42984	13989	79255	86600
Health	51164	28161	76716	87311
Home	33578	24163	77822	87221
Kids and Teens	12790	12790	78790	79091
News	48311	48337	76116	76182
Recreation	99150	45136	75715	96773
Reference	80513	48967	73782	73532
Regional	654467	9121	80904	90284
Science	75738	15132	79656	92201
Shopping	102067	73385	72101	94066
Society	183309	77882	45146	104355
Sports	74341	17092	79314	89360

Table 1: The number of Training and Testing Web documents in the top-level. (P=Positive, N=Negative)

Category Name	**Total**	**Training**		**Testing**
		P	N	P+N
Arts.Animation	16068	717	5962	138
Business.Accounting	1243	767	13854	105
Computers.Graphics	1568	1523	5621	138
Games.Gambling	3048	1797	1905	109
Health.Fitness	1118	963	4630	108
Home.Gardens	3178	2124	2927	120
News.Media	685	633	5945	83

Table 2: Training and Testing Samples in the second-level. (P=Positive, N=Negative)

Level	**miR**	**miP**	**miF_1**	**maF_1**	**error**
top	0.731	0.739	0.735	0.710	0.078
second	0.752	0.730	0.741	0.672	0.117

Table 3: Performance of classifiers in the top-level and the second-level using SVM classifiers.

Category Name	Topic number
Arts	505, 510, 527, 534, 545
Business	502, 514, 538, 542
Computers	501
Games	-
Health	504, 508, 509, 511, 524, 532, 537, 539, 540, 542, 543, 544, 549
Home	507, 514, 548
Kids and Teens	512, 515, 519, 525, 527, 539, 549
News	541
Recreation	503, 507, 519, 521, 529, 535, 547, 549
Reference	502, 547
Regional	-
Science	501, 502, 504, 513, 519, 522, 525, 528, 530, 542, 549, 550
Shopping	507, 508, 509, 511, 517, 519, 520, 522, 530, 532, 537, 539, 548, 549
Society	509, 510, 516, 517, 519, 520, 529, 534, 544, 545
Sports	506, 509, 548

Table 4: The categories of Topics.

Evaluating classifiers

After building the classifier in the training process, we obtain 15 top-level classifiers and 431 second-level classifiers. Then we test both levels classifiers with provided testing sets. To evaluate the performance of classifiers, we use the standard precision (P), recall (R) and F_1 measures. Precision is the ratio of correct assignments by the system to the total number of the system's assignments. Recall is defined to be the ratio of correct assignments by the system to the total number of correct assignments. The F_1 measures equally weights between precision and recall in the following form :

$$F_1(R,P) = \frac{2RP}{R+P} \quad (3)$$

This measure can be computed in 2 ways, micro-averaging and macro-averaging. Micro-averaging can be calculated from global average in all categories. Macro-average can be calculated from individual calculation in each category first, and then average over categories. Table 3 shows the accuracy of our both top and second level classifiers.

Applying the classifiers with TREC10

We use the top-level classifiers to classify the WT10g collection and its 50 queries into 15 top-level categories: Arts, Business, Computers, Games, Health, Home, Kids_and_Teens, News, Recreation, Reference, Regional, Science, Shopping, Society and Sports. Some documents and topics cannot be classified in any category, while some can be classified in one or more categories. Categories of topics are summarized in Table 4. There are 7 topics that cannot be contained in any category i.e. 518, 523, 526, 531, 533, 536 and 546. Then we pass the result to the second-level classifier to classify them into 431 second-level categories. The documents that pass the top-level classifier in each category are classified by the second-level classifiers of that category. Table 5 shows an example of the result. Topic number 507 is in category Home, Recreation, and Shopping within the top-level. After being classified by the second-level classifier, topic 507 resides in Home&Customer_Information, Home&Homeowners, Recreation&Autos, Recreation&Climbing, Shopping&Office_Products, Shopping&Tobacco, and Shopping&Vehicles, which are still in the sub-category of Home, Recreation, and Shopping. We show the amount of documents in the top and the second level category in Figure 1 and 2 respectively. The X axis in figure 1 represents the 15 top-level categories ranging from Arts to Sports, while the X axis in figure 2 represents all 431 sub-categories in the second-level.

Topic number	Category of topic
507	Home.Consumer_Information
	Home.Homeowners
	Recreation.Autos
	Recreation.Climbing
	Shopping.Office_Products
	Shopping.Tobacco
	Shopping.Vehicles

Table 5: The second-level category of topic 507

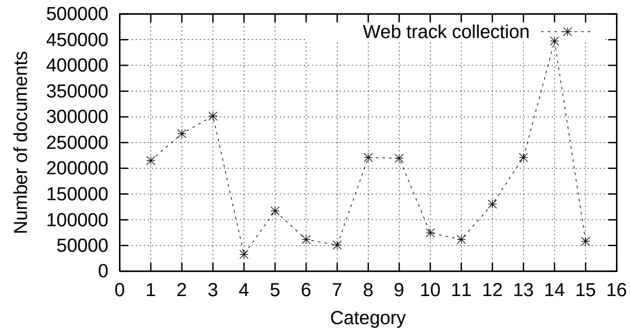

Figure 1: The number of documents in the top-level category.

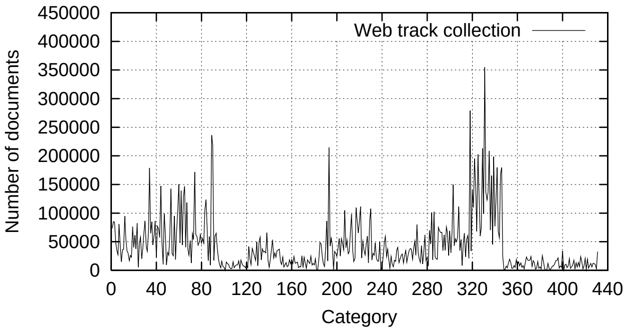

Figure 2: The number of documents in the second-level category.

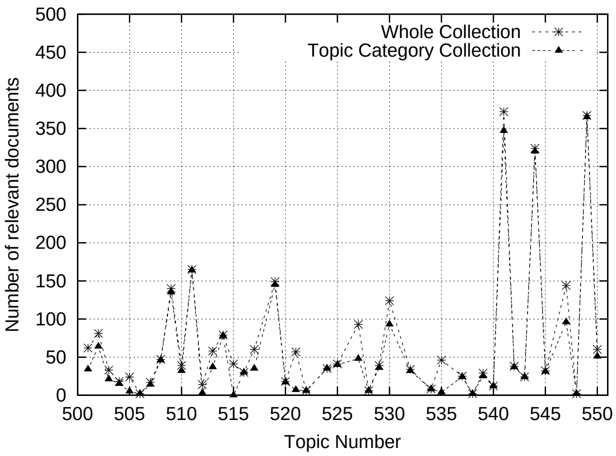

Figure 3: Comparison between relevant documents in the whole collection and the top-level topic category collection.

3.2 Experimental Setup and Results

We set the experiments into 3 phases: evaluating the assumption, searching in topic category collection and re-ranking score.

Phase I: Evaluating the assumption

In this sub-section we evaluate our assumption that "Most relevant documents are in the same categories as their query", by examining the number of relevant documents in topic category collection. Topic category collection of each query is defined as a set of documents that is in the same categories as its query.

Figure 3 provides the comparison between the total amount of relevant documents that can be retrieved from the whole collection and those from the top-level topic category collection. The X axis is the topic number. The Y axis is the number of relevant documents. For example, for the topic number 541, there are totally 372 relevant documents in the whole collection, and 347 relevant documents in the top-level topic category collection.

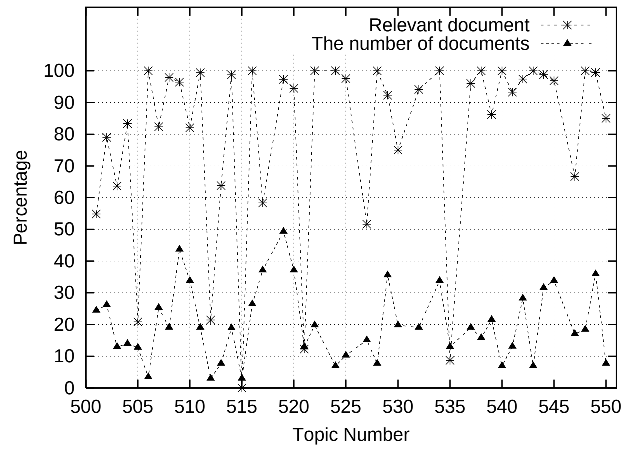

Figure 4: Percentage of relevant documents and number of documents in the top-level topic category collection.

Figure 4 shows the percentage of relevant documents and the percentage of the number of documents for each topic in the top-level topic category collection. For example, for the topic number 540, there are 100% of the total relevant documents and 6.93% of the whole documents being categorized in the top-level topic category collection. Figure 5 shows the comparison between the number of relevant documents for each topic in the top-level topic category collection and the second-level topic category collection, while figure 6 shows the comparison of the number of documents being categorized in both levels.

As we can see from figure 3 and 4, most relevant documents are categorized in their corresponding topic category collection. There are 12 topics that have the number of relevant documents in the topic category collection less than 80 percents. Moreover, the size of the topic category collection for each topic is less than 50 percents of the whole collection. This shows that our assumption is correct for most of the relevant documents and their corresponding topics. From figure 5 and 6, relevant documents in the top-level topic category collection are also found in the second-level topic category collection, while the number of documents in the second-level topic category collection decrease about 30 percents on average. This shows that our assumption in the second-level is still correct.

However, there are some topics in both levels that do not follow this assumption. The cause may be come from the inefficiency of the classifier we use. There are some topics and documents that are classified to the wrong category. For this reason the result does not as good as it should be.

Experiment	**Relevant**		**Average per Query**		**Average Precision**
	Doc	Ret	Searched Documents	Searching time (sec)	
Retrieve from whole documents	3363	2247	1692096 (100%)	20.5067	0.2088
Retrieve from top-level topic category collection	3363	2132	530332 (31.34%)	7.7179 (37.63%)	0.2032 (−2.7%)
Retrieve from second-level topic category collection	3363	2030	450912 (26.65%)	6.8697 (33.50%)	0.1997 (−4.4%)
Score Re-ranking	3363	2300	1692096 (100%)	-	0.2163 (+3.6%)

Table 6: Average precision concluded from the experiments (50 Queries).

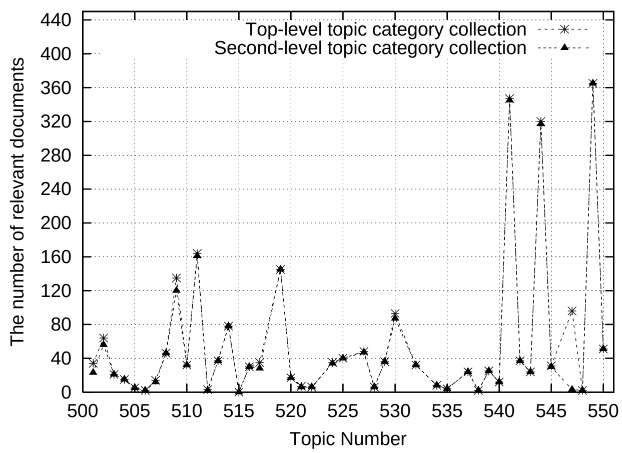

Figure 5: Comparison between relevant documents in the top-level and the second-level topic category collection.

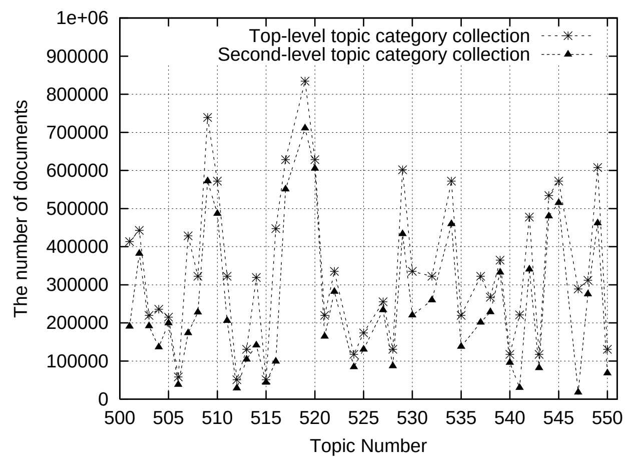

Figure 6: Comparison between the number of documents in the top-level and the second-level topic category collection.

Phase II: Searching in topic category collection

In this phase, we study the average precision of searching in both levels of topic category collection. We retrieve top 1000 documents for each query. Table 6 show the result of this phase comparing with the searching result from the whole collection. From Table 6, the average precision (0.2032) from searching only in the top-level topic category collection is almost the same as the average precision obtaining from searching in the whole collection (0.2088), while the number of searching documents and the searching time are only 31.34% and 37.63% of the whole. In case of the second-level, the average precision (0.1997) decreases a little bit, while the number of searching documents and the searching time are only 26.65% and 33.50% of the whole.

Phase III: Re-ranking Score

The experiments from re-ranking phase use the prediction value from the SVM classifiers. We retrieve 5000 documents for each query and re-rank the score of each retrieved document depending on its prediction value using the re-ranking equation(5) below and then we keep top 1000 documents for evaluation. The parameters of the equation are the prediction value, the constants, and the original score. In current experiments we can increase the precision a little bit. The result shows that our re-ranking equation does not take much effect so we must still look for better parameters of equation. For this experiment we consider only the top-level topic category documents. The re-ranking equation is:

$$Newscore = orig\_score + 0.5 \times orig\_score \times i \quad (4)$$

$$i = \frac{predict\_value}{2.75}; (i > 1) \rightarrow (i = 1) \quad (5)$$

The prediction value is calculated as the following. For example, query number 1 is in category A and B with prediction value 0.1 and 0.3. Document 1 is in category A, B, and C. Document 2 is in category A and C. Document 3 is in category C. The prediction

Experiment	Relevant document	Relevant retrieved	Average precision
HomePage Finding Task with Query Expansion	252	120	0.1902

Table 7: Final result of HomePage Finding Task with Query Expansion.

value in query 1 of document 1 is 0.4, document 2 is 0.1 and document 3 is 0. The number of 0.5 and 2.75 in equation (4) and (5) are concluded from the experiments, 0.5 controls the max increasing score, 2.75 is the reference value. If our prediction value exceeds the reference value, the total point (half of original score) will be increased.

Table 6 shows that re-ranking method can increase the fraction of precision from 0.2088 to 0.2163 (+3.6%). However, this re-ranking method is in preliminary step, we are looking for better re-ranking equation to get more retrieval accuracy.

4 Homepage Finding Task

Since topics in homepage finding task have very few words, we then apply an query expansion technique [16] to add some more useful words to them. We first remove stop words and stem the rest of the homepage finding topics and use them as our unexpanded queries. Our query expansion method has been divided into 3 steps described in the next paragraph. After these steps, we obtain 5 types of queries: I, II, III, IV and V, respectively. The query expansion type which gives the best result for homepage finding training set is chosen to expand the homepage finding topics.

Step I, we send the unexpanded queries to the Google search engine and keep the top 20 search results. If the query length is more than 10 words, we break that query into multiple word-segments (each segment has at least 9 words), send every word-segment to the Google search engine and keep the top 20 search results of all word-segments. We then intersect each search result of the word-segments to obtain the final 20 search results.

Step II, we remove stop words and stem the rest of the search results from step I and find document frequency of all words. If document frequency of any word is more than 5, we will use that words to expand the query to be searched in the homepage finding collection. This query is called Type I query. For those words whose document frequency are less than or equal to 5, they will be sorted by their document frequency, the first highest document frequency is used to build the Type II query, the second highest document frequency is used to build the Type III query, and so on.

Step III, for the Type II query, we add the first highest document frequency word that we obtain from step II above to the unexpanded query and resend that query to the Google search engine to get the top 20 search results. For these 20 search results, we repeat the step II, i.e. remove stop words and stem the rest and keep only words whose document frequency are more than 5 to add to the unexpanded query to build the Type II query. For the Type III to Type V query we repeat the same process as we do for the Type II query described in the beginning of this paragraph.

After we tested 5 Types of the expanded queries with homepage finding training set, we have found that the Type IV query provided the best result. We then apply the Type IV query to expand all the homepage finding queries, and use those expanded queries to search in the homepage finding collection. Table 7 concludes the final results.

5 Conclusion

In our Kasetsart TREC-10 experiments this year, we change indexing and testing tool from SMART version 11.0 to WORMS, our own retrieval engine, and use the Okapi weighting scheme in both web ad hoc and homepage finding tasks instead of the pivoted length normalized weighting scheme that we used in TREC-9 experiments last year. We also propose a novel retrieval approach using text categorization to improve retrieval effectiveness of the TREC-10 web ad hoc task. After we re-rank the resulting score, we get a little bit of precision increased. By searching relevant documents only in the top-level topic category collection, we obtain as good average precision as searching in the whole document collection, while the number of searching documents and the searching time reduce to 31.34% and 37.63% of the whole.

In the homepage finding task, we apply a query expansion method and using the Google search engine to find more words to be added into the homepage finding topics. We found that the result is quite promising.

Acknowledgement

We would like to thank all MIKE staffs, especially Hong, Wit, A, Banana, for their programming support and working spirit. We also thank Mr. Somsak Sriprayoonsakul from the Parallel Research Group for good support in our work.

References

[1] Google search engine. http://www.google.com.

[2] Open directory project. http://dmoz.org/.

[3] C. B. A. Singhal and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM-SIGIR Conference. ACM Press*, pages 412–420, 1996.

[4] R. Baeza-Yates and B. Riberiro-neto. *Modern Information Retrieval*, chapter 7, pages 167–168. Addison-Wesley, 1999.

[5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[6] D. Filo and J. Yang. Yahoo home page. http://www.yahoo.com.

[7] A. Inc. Altavista search index. http://www.altavista.digital.com.

[8] T. Joachims. Making large-scale svm learning practical. *In Advances in Kernel Methods. MIT Press*, 1998.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *European Conference on Machine Learning (ECML)*, 1998.

[10] Y. Labrou and T. Finin. Yahoo! as an ontology - using yahoo! categories to describe documents to describe documents. *In Proceedings of CIKM'99*, pages 180–187, Oct. 1999.

[11] P. Norasetsathaporn and A. Rungsawang. Kasetsart university trec-9 experiments. *In The Ninth Text Retrieval Conference (TREC-9), NIST Special Publication 500-249*, 2000.

[12] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical report, AIM-1602, 1997.

[13] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-2. *In The Second Text REtrieval Conference (TREC-2), NIST Special Special Publication 500-215*, pages 21–34, August-September 1993.

[14] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. *In The Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-226*, November 1994.

[15] A. Rungsawang, P. Uthayopas, M. Lertprasertkul, P. Ingongngam, and A. Laohakanniyom. Worms: A high-performance text retrieval prototype. *HPC-ASIA The Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region*, 2001.

[16] A. Sugiura and O. Etzioni. Query routing for web search engines. In *The Proceedings 9 th International World Wide Web Conference*, pages 412–420, May 2000.

[17] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997.