Experiments on Automatic Web Page Categorization for IR system

Hisao Mase
Department of Computer Science, Stanford University
Gates RM418, Stanford, CA 94305, U.S.A.
mase@db.stanford.edu

Abstract

This paper describes keyword-based Web page categorization. Our goal is to embed our categorization technique into information retrieval (IR) systems to facilitate the end-users' search task. In such systems, search results must be categorized faster, while keeping accuracy high. Our categorization system uses a knowledge base (KB) to assign categories to Web pages. The KB contains a set of characteristic keywords with weights by category, and is automatically generated from training texts. With the keyword-based approach, the algorithms to extract keywords and assign weights to them should be considered, because the algorithms affect strongly both categorization accuracy and processing speed. Furthermore, we must take two characteristics of Web pages into account: (1) the text length is very variable, which makes it harder to use statistics such as word frequency to calculate keyword weights, and (2) a huge number of distinct words are used, which makes the KB bigger and therefore processing speed lower. We propose five kinds of methods to normalize word frequency distribution for higher categorization accuracy, and three kinds of methods to filter out non-important words from the KB for faster processing. We performed experiments to compare these methods from viewpoints of both accuracy and KB size. We used 15 categories, 10,311 Web pages for KB generation and 939 pages for testing. The results show that the KBs with various accuracy values and sizes could be generated by applying our methods and that it is possible for end-users to select the most appropriate KB according to their preferences in accuracy and speed.

1. Introduction

It is now possible to obtain all kinds of information via the Internet. Many types of search engines are available to do the search. As accessible information increases, however, it has become more and more difficult to obtain the information rapidly and easily. One solution to this problem is to categorize the documents according to their topics beforehand or in real time. Some search engines such as Yahoo! (Yahoo! 1995) adopt category-based search. However, it is a time-consuming task for us to categorize a huge number of pages correctly, because we must read and analyze each of them.

Many researchers have been working on automatic categorization for special types of documents, such as newspaper articles (Hayes 1990) and patent documents (Mase 1996). Infoseek adopts neural network technology to categorize millions of Web pages automatically (Infoseek 1996). Some categorization systems use pre-defined categories (Mase 1996), and others use dynamically generated clusters (Sahami 1998, Iwayama 1995).

In information retrieval (IR) systems such as Infoseek and Yahoo!, category hierarchies do not always fit end-users' search purposes. If search results could be categorized in real time according to end-users' own categories, the category information would be more useful to facilitate their search task. In order to apply real time categorization technique to IR systems, however, categorization algorithms for high accuracy and reasonable processing speed would be required.

We have used a categorization knowledge base (KB) to assign categories automatically to query texts [Mase 1996]. The KB contains a set of characteristic keywords with weights by category and is automatically generated from training texts by using word frequency. With the keyword-based approach, however, the algorithms to extract keywords and assign weights to them should be considered, because the algorithms affect strongly both accuracy and processing speed.

Furthermore, since we focus on Web pages as target texts, we must take the following two characteristics of Web pages into account:

(1) *Text length is very variable.*

Some pages include short texts and some include extremely long texts. This makes it harder to use word frequency to calculate keyword weights, because the word frequency depends strongly on text length and the training pages with long texts might dominate the KB.

(2) *A huge number of distinct words are used.*

Since there is no restriction on word usage, too many kinds of words are used, including proper noun words and misspelled words. Since the KB contains all the keywords in training texts, the KB size will be bigger if the mechanism to distinguish important keywords from non-important words does not exist. The bigger KB causes a lower processing speed in keyword matching.

In this paper, we propose five methods to calculate keyword weights based on word frequency, especially to normalize keyword frequency (weights) distribution for

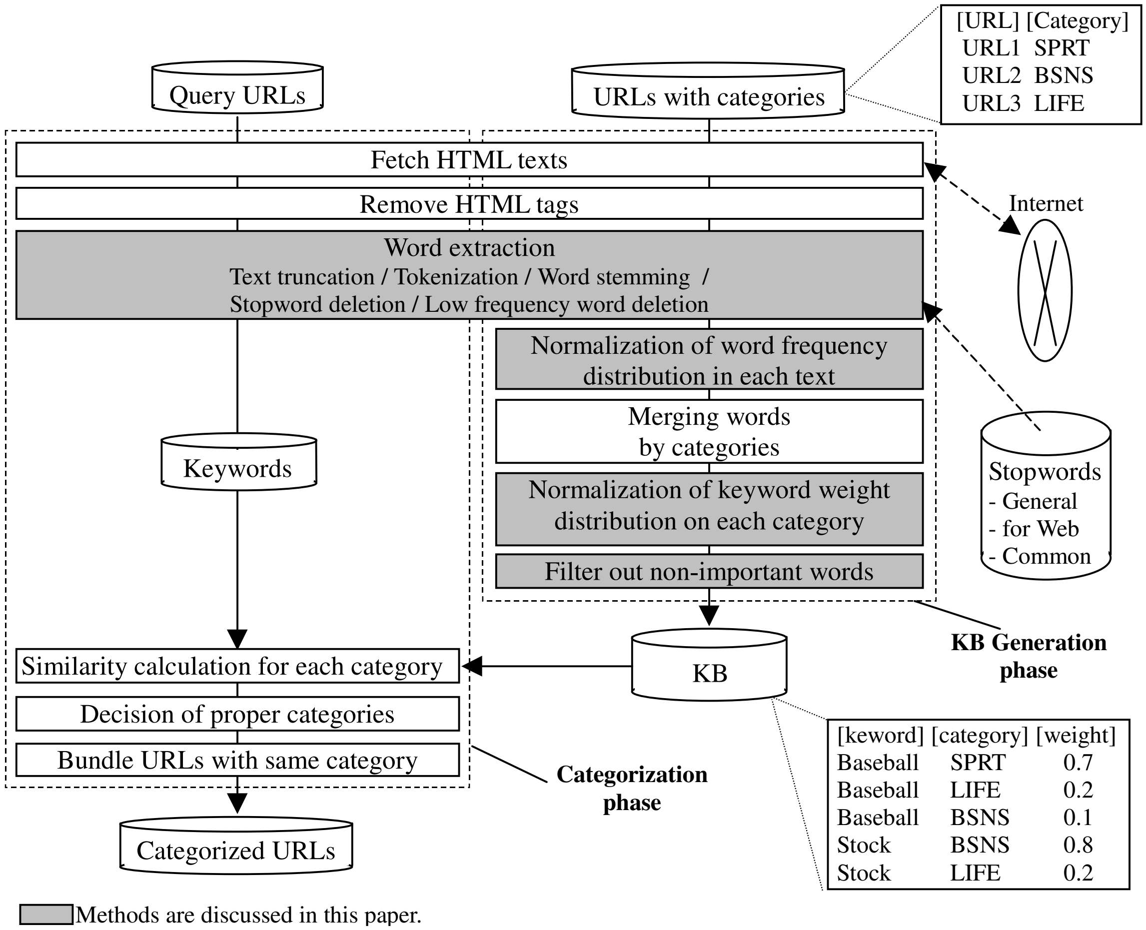


Figure 1: Processing flow of the system.

higher accuracy (these solve the problem described above in (1)). We also propose three kinds of methods to filter out non-important words from the KB for faster processing (these solve the problem described in (2)). We performed experiments to compare our methods from viewpoints of both accuracy and KB size. In these experiments, we used 15 categories, 10,311 Web pages for KB generation and 939 pages for testing.

Section 2 describes the overview of our categorization system. Section 3 describes our methods of normalization and word filtering. In Section 4 we compare these methods through experiments.

2. Overview of our Categorization System

2.1 Premises of our System

Our system has the following restrictions on categorization:

(1) *Categories must be pre-defined.*

Unique category names must be described beforehand in a particular text file by end-users or system administrators.

(2) *The category domains should be mutually exclusive.*

This means that two categories share few domains. If the domain of category-A is a subset of that of category-B, then it is difficult to categorize documents into category-A by keyword-based approaches.

(3) *Training texts must be prepared for KB generation.*

A training text set must be prepared, because our system extracts keywords on each category from them. Each training text must have one or more appropriate categories. Although it might be time-consuming to collect training data from end-users, it is much more difficult for them to define enough kinds of keywords for each category by hand.

(4) *There must be enough keywords in a text.*

If there is no text or few words in a query text, it is impossible for our system to categorize it.

2.2 Processing Flow of our System

Figure 1 shows the processing flow of our Web page categorization system. It consists of two phases: KB generation phase and categorization phase.

2.2.1 KB Generation

KB generation is executed in batch mode. It is performed by the following modules, some of which are shared with the categorization phase:

[1] *Preprocessing*

Our system gets a URL list with categories as input. First, our system fetches HTML texts and removes HTML tags in them as preprocessing. The system calls a utility of the programming language Python (Python 1998) to fetch HTML texts. We set a time-out function in the fetching process for faster processing. That is, if the fetching program cannot connect a target Web server within predefined time, the system terminates the fetching of this HTML text. Currently, fetching processing is executed sequentially.

The HTML tag deletion algorithm is simple. It removes the character strings surrounded by “<“ and “>” by checking each character in the text.

[2] *Word extraction*

This module consists of five sub-modules: text truncation, tokenization, word stemming, stopword deletion and low frequency word deletion. These processes other than tokenization are optional.

For the purpose of fast processing, text truncation extracts the first N characters from the text and removes the rest of the text. N is a tunable parameter.

Tokenization extracts words from the text. Our system does not use word dictionaries, because it takes more time to access them. In order to exclude as many noise words as possible, only the words that meet all of the following conditions are extracted as keyword candidates:

- It begins with a letter a-z or A-Z.
- It consists of letters, digits, hyphens and underscores.
- The length is less than 20.

All capitalized letters are translated into lower case.

Word stemming translates the extracted words to their stems so that the word frequency is counted more correctly. We adopt a free stemming program (Porter 1980). By stemming words, for example, the words “study”, “studies”, and “studied” are regarded as the stemmed word “studi”.

Stopword deletion removes predefined non-important words from the word set. We define three kinds of stopword lists: “general stopwords”, “stopword for Web pages” and “common words”. General stopwords are independent of the domains or formats of the target texts, including “the”, “of”, “is”, etc. We prepared 446 general stopwords by hand from a word dictionary. Stopwords for Web pages include the words appearing in most Web pages, such as “home” and “html”. We defined 35 stopwords for Web pages by hand. Common words are the words appearing over many categories. Whether one word is a common word or not is strongly dependent on the domains and category definition. Our system extracted 330 common words from the training texts in the experiments described in section 4.

Low frequency word deletion cuts off words with relatively low frequency of occurrence in a text for faster processing. The details for this algorithm are described in Section 3.2.

[3] *Normalization of word frequency distribution on each text*

As described in Section 1, the text length of a Web page is quite variable. As a result, the pages with longer texts include more kinds of words with higher frequencies, as shown in [step 1] of Figure 2. In other words, the pages with longer texts dominate the KB. In order to solve this problem, we should consider the normalization of the word frequency distribution on each text so that we can treat the texts with different lengths equally, as shown in [step 2] of Figure 2. We propose five kinds of methods to normalize word frequency in each page in Section 3.1.

[4] *Merging words by category*

All training texts are given their corresponding categories. All words appearing in the texts with the same category are merged. Then the weights of the same word are summed up as shown in [step 3] of Figure 2.

[5] *Normalization of keyword weight distribution on each category*

The number of training texts for each category might be different. If a word appears in more texts with a particular category, the weight of the word is higher. We should normalize distribution of keyword weights on each category, as shown in [step 4] of Figure 2. We propose five kinds of methods in Section 3.1.

[6] *Filtering out non-important words*

This module cuts off non-important words in the KB as noise words. We propose three kinds of word filters, plus their combinations, in Section 3.1. The filters use the weight distribution of a keyword to decide whether a word should be removed or not. Reducing the KB size not only makes the word matching process faster but also facilitates KB maintenance by end-users or system administrators. Notice that word removal in this module is different from that in the word extraction module: the former cuts off words in the KB, while the latter cuts off words from a set of words extracted from a text.

[7] *KB structure*

The KB is a set of keywords with their weights by category. Each record in the KB consists of category name, keyword string and its weight in that category, as shown in Figure 1.

2.2.2 Categorization of New Pages

In the categorization phase, our system gets a set of URLs without categories as input from an IR system that requires the organization of search results, such as Sensemaker (Baldonado 1997).

[1] *Preprocessing and word extraction*

Our system uses the same modules as those used in the KB generation phase (see Section 2.2.1).

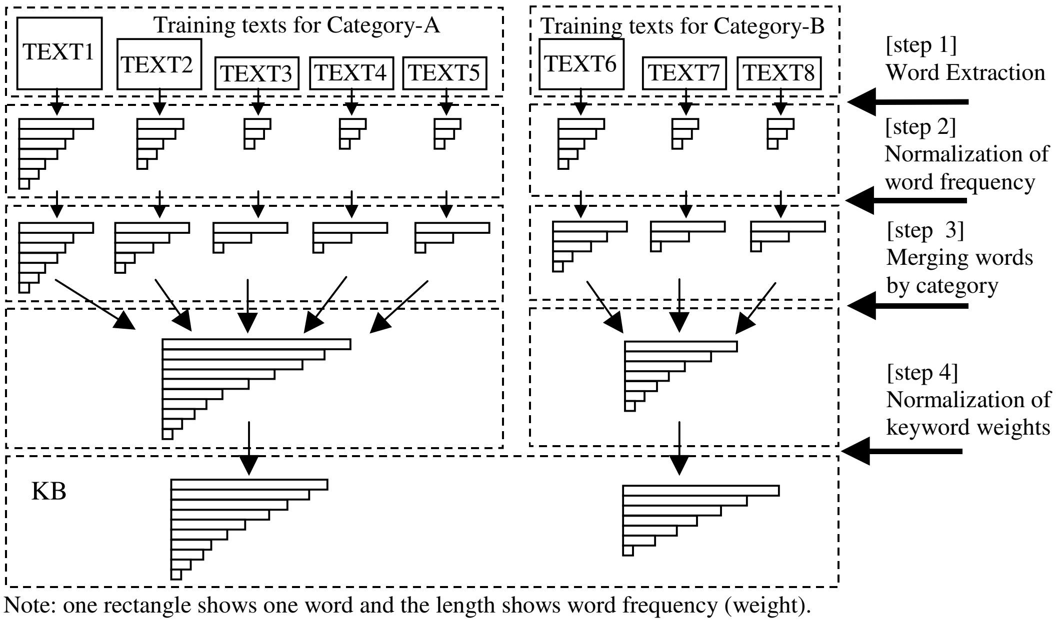


Figure 2: Mechanism of normalization for KB generation.

[2] *Similarity calculation for each category*

We use the following "dot product" formula to calculate similarity to each category:

$$S(i)= \Sigma s(i, j) \qquad (j=1,2,\ldots,n)$$
$$s(i, j)=W(j)*(w(i, j)/\Sigma w(k,j)) \qquad (k=1,2,\ldots,m)$$

where $S(i)$ is a similarity value of category-i, $s(i,j)$ is a similarity of category-i on a particular keyword-j in a query text, n is a number of distinct keywords extracted from a query text, $W(j)$ is a weight of keyword-j in a query text, $w(i,j)$ is a keyword weight of keyword-j on category-i in the KB, and m is a number of categories. This formula has the following characteristics:

- The higher the weight of a keyword-j extracted from a query text ($W(j)$) is, the higher the similarity value on that keyword ($s(i,j)$) is.
- The higher the relative weight of a keyword-j in a category-i ($w(i, j)/\Sigma w(k,j)$) is, the higher the similarity value on that keyword on that category ($s(i,j)$) is.
- When a keyword-j appears over more categories, the similarity value of that keyword on that category ($s(i,j)$) is lower, because the value $\Sigma w(k,j)$ is higher.

[3] *Decision of proper categories*

The calculated similarity values are sorted and the top N categories are assigned to the URL. It is difficult to decide the value of N automatically. We will discuss multiple category assignment in Section 4.5.2. In the experiments in Section 4, N is equal to 1.

[4] *Bundle URLs with the same category*

URLs with the same category are bundled together. The bundle is a set of URLs with the same category. One bundle corresponds exactly to one category. The URLs with multiple categories may belong to multiple bundles. Finally, our system outputs these bundles.

3. Methods for higher accuracy and faster processing

As described in Section 1, keyword extraction and weight assignment procedures affect strongly both accuracy and processing speed. In this section we propose methods for higher accuracy and faster processing.

3.1 KB Generation Phase

Since the KB is generated in batch mode, processing time is not a major concern. Thus, we propose methods on the following three kinds of processing only for higher accuracy:

[1] *How to normalize word frequency distribution in each text*

Our system is now given a set of pairs of a keyword and its frequency. The goal of this normalization is to assign a keyword weight which is independent of the length of the HTML text in which the keyword appears. We propose the following five normalization methods.

[Method-1] *Square root operator*

This method is simple. The square root operator is applied to each value of word frequency as shown in Figure 3[1]. It reduces the difference of the word frequency.

[Method-2] *Linear mapping*

Method-2, 3 and 4 focus on the maximum value (VMAX) and the minimum value (VMIN) of word

Keyword	Original word set	[1] Square root	[2] Linear mapping	[3] Mapping using quadratic function	[4] Mapping using square root function	[5] Relative value
KW1	10	3.16	100	100	100	0.20
KW2	9	3.00	89.0	79.2	94.3	0.18
KW3	8	2.83	78.0	60.9	88.3	0.16
KW4	7	2.65	67.0	45.0	81.8	0.14
KW5	5	2.24	45.0	20.6	67.0	0.10
KW6	4	2.00	34.0	12.0	58.2	0.08
KW7	3	1.73	23.0	5.89	47.7	0.06
KW8	2	1.41	12.0	2.22	34.0	0.04
KW9	1	1.00	1.00	1.00	1.00	0.02
KW10	1	1.00	1.00	1.00	1.00	0.02

VMAX=10
VMIN=1
Total # of words=50

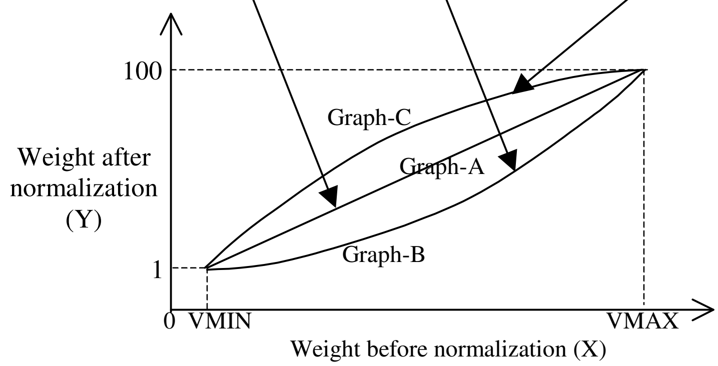


Figure 3: Five methods for Normalization of distribution of word frequency.

frequency in a text. All values are mapped into a fixed scale of 1 to 100. VMAX is mapped into 100, and VMIN to 1. The difference among these three methods is how to map values other than VMAX and VMIN. In Method-2, the value is mapped linearly as shown in Figure 3[2]. That is, the value X is mapped into Y, using the following formula which shows graph-A in Figure 3:

$$Y=1+(100-1)*(X-VMIN)/(VMAX-VMIN)$$

[Method-3] *Mapping using quadratic function*

In this method, the value is mapped according to a quadratic function as shown in Figure 3[3]. The value X is mapped into Y, using the following formula corresponding to graph-B in Figure 3:

$$Y=1+(100-1)*(X-VMIN)^2/(VMAX-VMIN)^2$$

Notice that the normalized value by this formula is lower than the one by Method-2.

[Method-4] *Mapping using square root function*

In this method, the value is mapped based on square root function as shown in Figure 3[4]. The value X is mapped into Y, using the following formula, corresponding to graph-C in Figure 3:

$$Y=1+(100-1)*sqrt(X-VMIN)/sqrt(VMAX-VMIN)$$

Notice that the normalized value by this formula is higher than the one by Method-2 or 3.

[Method-5] *Relative values to total number of words*

In this method, the total number of words in a text is calculated. Then, the normalized value is calculated by dividing the original word frequency value by the total number, as shown in Figure 3[5]. Notice that if the total number of words is small, the normalized value is high even though its frequency is low.

[2] *How to normalize keyword weight distribution on each category*

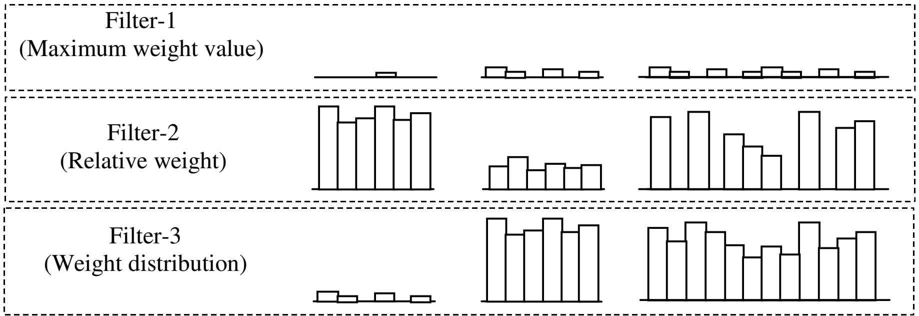
Our system is now given a set of pairs of keyword and its “normalized” frequency. Since all words in the texts with the same category are merged and the normalized frequency values of the same words are summed up (see [step 3] of Figure 2), the keyword weight on a category with more training texts than other categories is higher. The goal of this normalization is to assign a keyword weight which is independent of the number of the training texts. We adopt the same five methods as those mentioned above [1].

[3] *How to filter out non-important words*

Our system is now given a set of triples of category-i, keyword-j and its normalized weight (w(i,j)) as the KB. However, this KB includes many non-important words to be removed, which makes the KB bigger and therefore the processing speed lower. Thus we propose the following three kinds of filters for KB size reduction:

[Filter-1] *Maximum value of weights*

This filter focuses on the maximum value of the normalized weights in each keyword. The words with low weight should be removed from the KB, because our system cannot obtain enough statistic data on those words to decide whether they are important keywords or



Note1: The length of one rectangle shows a weight of one category.

Figure 4: Distribution patterns of weights of the word which each filter cuts off.

not. With this filter, the word with lower maximum values over normalized weights by category in the word than a threshold (tunable) is removed from the KB as shown in Figure 4.

[Filter-2] *Relative value of weights*

This filter focuses on the relative value of the weights on a keyword, which is similar to Method-5 mentioned above [1]. First, the total of the weights for a keyword is calculated. Then each weight is normalized by dividing the weight by the total. The word with lower maximum value over the relative weights for the word than a threshold (tunable) is removed from the KB. This filter is useful to remove keywords with equal weights by category or those appearing over many categories.

[Filter-3] *Distribution of weights on a keyword*

This filter focuses on the "variance" of the distribution of the weights on a keyword. The important keywords for categorization should have significantly higher weights for the category they identify. Thus, the variance of the distribution of the weights is also higher. However, the variance value is higher if the average value of the weights is higher. Thus, we calculate a "normalized" variance by dividing the original variance by the average value of the weights. That is, it is calculated by the following formula:

$$nvar = var/m$$
$$var = \Sigma((w(k,j)-m)*(w(k,j)-m))/n \quad (k=1,\ldots,n)$$
$$m = \Sigma w(k,j)/n \quad (k=1,\ldots,n)$$

where, nvar is "normalized" variance, var is variance, m is the average value of keyword weights and n is the number of category. This filter removes the words whose "normalized" variance is lower than a threshold (tunable), as shown in Figure 4.

In these three filters, since we use thresholds to decide non-important words, our system is able to generate KBs with various sizes and accuracy values by tuning these thresholds. The best combination of them depends on texts, category definition and end-users' preferences.

3.2 Categorization Phase

Since faster processing is required in the categorization phase, it is important to develop good techniques for it. In addition to the idea we have discussed, we have the following:

[1] *How to truncate the query text*

Since the text length of Web pages varies widely, it takes much time if an extremely long page is included in a search result. In order to improve the speed for such a page, text truncation is most useful, because it takes little time to truncate the text. Thus, our system only considers the first N characters of the text. Text truncation also reduces the number of distinct keywords, which facilitates faster processing.

[2] *How to reduce the words extracted from the query text*

The words with lower frequency in a query do not significantly impact the category decision. Thus, removing such keywords improves processing speed, because the time to match with the KB is reduced. Our method focuses on the frequency of occurrence of each word and the total number of distinct words in a query. That is, we remove words with frequencies lower than the following value F:

$$F=[\text{Total number of distinct words}]/M$$

where M is tunable parameter. If M is set lower (the threshold F is higher), the number of extracted words is lower and the processing speed is higher.

4. Experiments

We performed experiments to compare the methods described in Section 3.

4.1 Data Collection

[1] *Category definition*

We used Infoseek categories (the 15 categories shown in Figure 5, together with their corresponding 240 subcategories) called "Channel" (Infoseek 1995).

[2] *HTML text data*

(a) *Training texts*

We collected a maximum of 100 URLs from each of the 240 Infoseek subcategory pages as training text data.

Figure 5: Infoseek 15 categories (as of January, 1998).

Automotive (AUTO)	Business (BSNS)	Computer (COMP)	Careers (CRER)	Entertainment (ENTR)
Personal Finance (FINC)	Health (HLTH)	Internet (INTR)	Kids & Family (KIDS)	Good-Life (LIFE)
News (NEWS)	Real Estate (REAL)	Shopping (SHOP)	Sports (SPRT)	Travel (TRVL)

Note: The string in parenthesis is abbreviation of the category used in other figures.

Figure 6: Number of training texts and test texts.

Category	Number of training texts	Number of test texts	Category	Number of training texts	Number of test texts	Category	Number of training texts	Number of test texts
AUTO	551	24	FINC	695	75	NEWS	568	38
BSNS	1079	70	HLTH	991	90	REAL	196	0
COMP	963	89	INTR	814	62	SHOP	754	66
CRER	426	39	KIDS	508	18	SPRT	1463	176
ENTR	1128	82	LIFE	884	53	TRVL	559	57

Figure 7: List of experiments.

#	KB generation methods					Word extraction from test texts	
	Stemming	Stopword deletion	Word frequency normalization	Keyword weight normalization	Word filters	Text truncation	Low frequent word deletion
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	Applied	N/A	N/A	N/A	N/A	N/A	N/A
3	Applied	Applied	N/A	N/A	N/A	N/A	N/A
4-1	Applied	Applied	Method-1	N/A	N/A	N/A	N/A
4-2	Applied	Applied	Method-2	N/A	N/A	N/A	N/A
4-3	Applied	Applied	Method-3	N/A	N/A	N/A	N/A
4-4	Applied	Applied	Method-4	N/A	N/A	N/A	N/A
4-5	Applied	Applied	Method-5	N/A	N/A	N/A	N/A
5-1	Applied	Applied	Best method	Method-1	N/A	N/A	N/A
5-2	Applied	Applied	Best method	Method-2	N/A	N/A	N/A
6-1	Applied	Applied	Best method	Best method	Filter-1	N/A	N/A
6-2	Applied	Applied	Best method	Best method	Filter-2	N/A	N/A
6-3	Applied	Applied	Best method	Best method	Filter-3	N/A	N/A
6-4	Applied	Applied	Best method	Best method	Filter-1 and 2	N/A	N/A
6-5	Applied	Applied	Best method	Best method	Filter-1 and 3	N/A	N/A
6-6	Applied	Applied	Best method	Best method	Filter-2 and 3	N/A	N/A
6-7	Applied	Applied	Best method	Best method	Filter-1, 2 and 3	N/A	N/A
7	Applied	Applied	Best method	Best method	Best method	Applied	N/A
8	Applied	Applied	Best method	Best method	Best method	N/A	Applied
9	Applied	Applied	Best method	Best method	Best method	Applied	Applied

As a result, we fetched 10,311 HTML texts.

(b) *Test texts*

As test data, we also collected 939 distinct HTML texts from the 240 Infoseek subcategory pages. None of these documents are included in the training data set. Each of these texts has only one category to be assigned. Figure 6 shows the number of training and test texts by category.

4.2 Experiment Methods

We performed nine kinds of experiments, as shown in Figure 7. In Experiment 1, we do not use any techniques we have discussed in this paper. That is, all words in the training texts are used to generate the KB. In Experiment 2, we apply stemming to words extracted from the training texts. In Experiment 3, we add stopword deletion. In Experiment 4, we compare our five methods to normalize the keyword frequency distribution. In Experiment 5, we compare our five methods to normalize keyword weight distribution in each category. In Experiment 6, we apply three kinds of word filters to the KB. In Experiment 7, text truncation is applied to query texts. In Experiment 8, low frequency word deletion is applied to words extracted from query texts. In Experiment 9, we apply both text truncation and low frequent word deletion.

4.3 Measures of Evaluation

We use three kinds of measures to evaluate our methods: categorization accuracy, standard deviation of accuracy values by category, and KB size.

Each of the test texts has only one correct category (its Infoseek category), and our system assigns only one category to each test text. We define categorization accuracy as the fraction of the test texts to which our system assigned the correct category.

We also take the distribution of accuracy values by category into account. We believe that the categorization system is not good if the accuracy values of particular categories are low, even though those of other categories are high. Thus we evaluate the standard deviation (SD) of the distribution of accuracy values by category. The SD is calculated as follows:

$$SD = \mathrm{sqrt}(var)$$
$$var = \Sigma((A_i - m)*(A_i - m))/n$$
$$m = \Sigma A_i/n \qquad (i=1,\ldots,n)$$

where var is the variance, m is the average value of accuracy, A_i is the accuracy of category-i, and n is the number of categories. The lower the SD value is, the better the system is. When we discuss the quality of categorization systems, we should consider both overall accuracy and the SD.

We use the KB size to evaluate the processing speed of categorization. In this paper, the KB size consists of a pair of the number of records and the number of distinct keywords. If the number of records is smaller, the speed of similarity calculation is higher. If the number of distinct keywords is smaller, the speed of keyword matching between keywords in a query text and those in the KB is higher. We do not consider in this paper the time to fetch texts and to extract words from them.

4.4 Results

[1] *Word stemming (#2 in Figure 7)*

Figure 8 shows the results of categorization with/without word stemming. Though stemming does not improve the accuracy, it reduces KB size by 20.7%. In both results, the SD is very high. Since the stemming algorithm we adopt might affect the accuracy, it would be interesting to evaluate it using other stemming programs. We use word stemming in all of the following experiments because it significantly reduces the KB size.

[2] *Stopword deletion (#3 in Figure 7)*

Figure 8 also shows the results of categorization with/without stopword deletion. Stopword deletion improves accuracy by 8.09% (78.27-70.18). The SD is also improved by 12.25 (33.62-21.37), although it is still high. Notice that stopword deletion does not reduce the KB size very much because we use a relatively small number of stopwords (670).

Figure 8: Results of stemming and stopword deletion (#1 to #3).

#	Distinct keywords in KB (kinds)	KB records (records)	Relative # of KB records (#1 is 1)	Accuracy	SD
1	213,184	512,786	1.000	71.03 %	31.59
2	181,708	406,714	0.793	70.18 %	33.62
3	181,112	398,084	0.776	78.27 %	21.37

Figure 9: Results of normalization on text (#4).

#	Method	Accuracy	SD
3	-	78.27 %	21.37
4-1	Square root operator	76.68 %	24.12
4-2	Linear mapping	81.36 %	20.58
4-3	Quadratic mapping	82.96 %	21.09
4-4	Square root mapping	80.09 %	20.02
4-5	Relative weight	83.60 %	18.08

Figure 10: Results of normalization on category (#5).

#	Method	Accuracy	SD
4-5	-	83.60 %	18.08
5-1	Square root operator	83.07 %	18.78
5-2	Linear mapping	85.52 %	10.76
5-3	Quadratic mapping	84.77 %	11.47
5-4	Square root mapping	84.35 %	11.14
5-5	Relative weight	84.13 %	11.02

[3] *How to normalize distribution of word frequency on each text (#4-1 to #4-5 in Figure 7)*

Figure 9 shows the results of categorization with/without normalization of word frequency distribution. Method-1, which uses the square root operator, has worse accuracy than the original scheme without normalization. The mapping using quadratic function provides the highest accuracy out of the three mapping methods. Method-5, which calculates relative value of word frequency, has the highest accuracy of the five. However, the accuracy differences are not major and SDs are still high. We use Method-5 in the following experiments. Note that this method does not reduce the KB size at all.

[4] *How to normalize distribution of keyword weights on each category (#5-1 to #5-5 in Figure 7)*

Figure 10 shows the results of categorization with/without normalization of distribution of keyword weights. The accuracy with linear mapping is a little higher than that with any other methods. This result is different from the result of Experiment #4. The remarkable thing is that SD as well as accuracy is improved using this normalization. This method does not reduce KB size, either.

Figure 11: Results of keyword filters(#6).

#	Threshold of Filter-1	Threshold of Filter-2	Threshold of Filter-3	Distinct keywords in KB (kinds)	KB records (records)	Relative # of KB records (#1 is 1.000)	Accuracy	SD
5-2	-	-	-	181,112	398,064	0.776	85.52 %	10.76
6-1-1	>1.00	-	-	88,860	280,776	0.548	84.77 %	12.35
6-1-2	1.05	-	-	37,467	187,626	0.366	83.71 %	15.83
6-1-3	1.10	-	-	23,199	146,987	0.287	82.96 %	16.90
6-1-4	1.20	-	-	14,247	110,163	0.215	82.53 %	17.14
6-1-5	1.50	-	-	7,247	69,554	0.136	82.11 %	17.56
6-1-6	2.00	-	-	4,113	45,322	0.088	82.11 %	17.79
6-1-7	3.00	-	-	2,154	26,876	0.052	81.36 %	19.65
6-1-8	5.00	-	-	1,087	14,522	0.028	80.30 %	20.63
6-1-9	10.00	-	-	380	5,383	0.010	78.27 %	20.59
6-1-10	20.00	-	-	149	2,109	0.004	75.40 %	20.59
6-1-11	30.00	-	-	88	1,274	0.002	73.38 %	19.54
6-1-12	50.00	-	-	46	670	0.001	60.81 %	30.14
6-2-1	-	0.10	-	179,032	370,482	0.722	85.62 %	10.57
6-2-2	-	0.15	-	172,290	300,087	0.585	85.94 %	10.46
6-2-3	-	0.20	-	167,856	265,229	0.517	85.84 %	9.67
6-2-4	-	0.25	-	163,932	241,992	0.472	86.16 %	9.20
6-2-5	-	0.30	-	158,500	218,156	0.425	85.62 %	9.43
6-2-6	-	0.35	-	149,042	188,111	0.367	84.88 %	9.78
6-2-7	-	0.40	-	147,459	182,467	0.356	83.92 %	10.23
6-2-8	-	0.45	-	147,133	181,018	0.353	83.60 %	10.60
6-2-9	-	0.50	-	146,982	180,159	0.351	82.96 %	10.40
6-3-1	-	-	0.50	172,719	301,077	0.587	85.52 %	11.01
6-3-2	-	-	0.80	159,548	228,659	0.446	85.62 %	10.80
6-3-3	-	-	0.90	121,677	140,335	0.274	85.20 %	11.25
6-3-4	-	-	1.00	9,519	22,204	0.043	82.11 %	18.10
6-3-5	-	-	2.00	841	6,278	0.012	80.62 %	19.45
6-3-6	-	-	5.00	220	2,575	0.005	76.89 %	20.97
6-3-7	-	-	10.00	105	1,422	0.003	76.04 %	19.87
6-3-8	-	-	20.00	60	855	0.002	67.20 %	25.81
6-3-9	-	-	30.00	34	490	0.001	59.00 %	27.37
6-3-10	-	-	50.00	12	165	0.0003	19.60 %	34.14
6-4-1	>1.00	0.25	-	71,929	126,020	0.246	85.30 %	10.81
6-4-2	1.05	0.25	-	23,584	51,677	0.101	84.56 %	13.08
6-4-3	1.10	0.25	-	12,177	31,313	0.061	83.28 %	16.00
6-4-4	1.20	0.25	-	6,406	20,371	0.040	82.64 %	17.02
6-4-5	1.50	0.25	-	2,871	13,251	0.026	81.79 %	17.76
6-5-1	>1.00	-	0.90	49,076	67,734	0.132	84.13 %	12.94
6-5-2	1.05	-	0.90	17,829	36,436	0.071	82.64 %	17.45
6-5-3	1.10	-	0.90	10,033	26,848	0.052	82.00 %	17.81
6-5-4	>1.00	-	1.00	9,519	22,204	0.043	82.11 %	18.10
6-5-5	1.05	-	1.00	9.519	22,204	0.043	82.11 %	18.10
6-5-6	1.10	-	1.00	7,807	20,492	0.040	82.11 %	18.10
6-6-1	-	0.25	0.90	121,324	135,301	0.264	85.94 %	9.53
6-6-2	-	0.25	1.00	9,236	18,112	0.035	82.43 %	16.92
6-7-1	>1.00	0.25	0.90	48,723	62,700	0.122	84.98 %	11.04

[5] *Filtering non-important words (#6-1 to #6-7 in Figure 7)*

Experiment #6-1-1 to #6-1-12 in Figure 11 show the results of categorization with/without Filter-1, which is based on the maximum value of the weights. The result shows that this filter can reduce KB size dramatically, while keeping the accuracy at more than 80%. When the maximum value threshold is 1.20 (#6-1-4), for example, the number of distinct keywords is reduced to 7.9% of the original number of words and the number of KB records to 21.5% of the original number of records. However, the SD is 17.14, which

Figure 12: Results of text truncation and low frequent word deletion (#7, #8 and #9).

#	Threshold of truncation (characters)	Threshold for low frequent word deletion	Average # of distinct keywords in query texts (kinds)	Accuracy	SD
6-7-1	-	-	274.4	84.98 %	11.04
7-1	10000	-	184.9	84.35 %	11.51
7-2	5000	-	132.2	83.49 %	12.47
7-3	3000	-	95.3	82.75 %	12.34
7-4	1000	-	38.1	79.55 %	15.02
8-1	-	200	75.5	83.92 %	12.95
8-2	-	150	56.2	83.71 %	13.39
8-3	-	100	35.8	82.11 %	14.90
8-4	-	50	14.1	80.09 %	16.79
9-1	5000	200	88.2	83.39 %	12.39
9-2	5000	150	57.5	82.75 %	13.50
9-3	5000	100	37.3	81.47 %	14.24
9-4	5000	50	14.4	79.34 %	16.52
9-5	3000	200	94.3	82.75 %	12.34
9-6	3000	150	77.0	82.53 %	12.30
9-7	3000	100	38.3	80.72 %	14.81
9-8	3000	50	15.2	77.96 %	16.31

is much higher than without this filter. The accuracy is reduced as the threshold value increases. The SD is also adversely affected.

Experiment #6-2-1 to #6-2-9 in Figure 11 show the results of categorization with/without Filter-2, which is based on relative keyword weight. Although this filter also reduces the KB size, the rate of reduction is much less than with Filter-1. However, this filter improves accuracy and reduces SD. When the threshold is 0.25, the accuracy is 0.64% higher than without this filter, and the SD is 9.20. These are the best values of all experiments. This result shows that this filter works very well.

Experiment #6-3-1 to #6-3-10 in Figure 11 show the results of categorization with/without Filter-3, which is based on the variance of the keyword weight distribution. When the threshold is less than or equal to 0.90, the accuracy and SD do not change, while the KB size, especially the number of KB records is reduced very much. When the threshold is 1.00, however, the accuracy becomes worse by 3.1%, and the SD is also worse by 6.9, while KB size is dramatically reduced. The reason is that there are many words appearing only a few times. If word-A appears only once in training texts and if its weight is 1.0, then the normalized variance is 0.94, according to the formula described in Section 3.1[3]. This shows that some of the words with lower frequency contribute to higher accuracy. This means that we should consider another method to retain these important keywords in the KB if we are to use this filter. For example, it might be useful to use HTML tag data to assign weights to words.

Experiments #6-4-1 to #6-7-1 in Figure 11 show the results of categorization using more than one filter. Since the number of combinations for the three thresholds is very high, we used only a few values for each filter, as shown in line #6-1-1 to #6-3-10 of Figure 11. These results show that some of the combinations improve KB size while keeping the accuracy almost constant. The result for Experiment #6-4-1, for example, shows that the accuracy is almost the same as that of #6-1-1 or #6-2-4 (where only one filter is used), and that the KB size is significantly reduced in comparison with #6-1-1 or #6-2-4. The result of #6-7-1, where all of three filters are used, provides an excellent balance between the competing factors. The accuracy is 84.94%, which is only 1.18% lower than the best accuracy value. The SD is 11.04, which is only 1.84 lower than the best one. The number of distinct keywords is reduced to 26.9% of those in Experiment #5-1 where no filters are used. The number of KB records is also reduced to 12.2% of those in Experiment #1 and 15.7% of those in Experiment #5-1. We use the KB resulting from this #6-7-1 set up.

[6] *Text truncation (#7-1 to #7-4)*

Experiments #7-1 to #7-4 in Figure 12 show the results of categorization with/without text truncation. We use 10000, 5000, 3000 and 1000 as the text threshold length. The percentage of the test texts that exceed (after HTML tag deletion) is 22.6%, 40.1%, 54.6% and 81.8%, respectively. When the threshold is 5000, the average number of keywords used for categorization is 48.2% of that used in experiment #6-7-1. Notice that the accuracy and SD are still good. The same thing can be said when the threshold is 3000.

Figure 13: KB selection for end-users' preferences.

End-user's requirement			The best KB			Accuracy	
Accuracy	SD	KB size	Exp. #	Distinct keywords	KB records	Accuracy	SD
best	best	-	6-2-4	163,932	241,992	86.16 %	9.20
85%	-	smaller	6-4-1	71,929	126,020	85.30 %	10.81
85%	10	smaller	6-6-1	121,324	135,301	85.94 %	9.53
80%	-	smaller	6-3-5	841	6,278	80.62 %	19.45
80%	15	smaller	6-4-2	23,584	51,677	84.56 %	13.08
75%	-	smaller	6-3-7	105	1,422	76.04 %	19.87
75%	18	smaller	6-4-5	2,871	13,251	81.79 %	17.76

[7] *Low frequent word deletion (#8-1 to #8-4)*

Experiment #8-1 to #8-4 in Figure 12 show the results of categorization with/without low frequency word deletion. We use 200, 150, 100 and 50 as the value M in the formula in Section 3.2[2]. The results show that the number of keywords is reduced dramatically, while the SD is a little worse.

[8] *The combination of text truncation and low frequency word deletion (#9-1 to #9-8)*

Experiment #9-1 to #9-8 in Figure 12 show the results of categorization with both text truncation and low frequent word deletion. We use 3000 and 5000 as the thresholds for truncation. Comparing the result of Experiment #9-1 with the result of #7-2, the average number of keywords is reduced to 66.7%, while keeping good accuracy and SD. This shows that the combination of text truncation and low frequency word deletion is effective for faster processing.

4.5 Discussion

4.5.1 KB Choice by End-users

The results of our experiments show that our methods are effective for high accuracy and KB size reduction. They also show that our system can generate KBs of different sizes by tuning the thresholds used in our methods. This means that our system can use the best KB for end-users to effectively categorize Web pages. For example, if a user requires the highest possible accuracy and if s/he does not care about the processing speed (KB size), then our system should use the KB used in Experiment #6-2-4. If s/he requires an accuracy of more than 80%, a SD of less than 15 and the fastest speed (the lowest KB size), then our system should use the KB of #6-4-2, and it should also apply text truncation and low frequent word deletion. We believe that it is important for IR systems to accept users' preferences in accuracy and processing speed. Figure 13 shows other examples of end-users' preferences and the corresponding KB choices.

Notice that the best threshold values depend on the text types (Web pages, patent documents, news articles, etc.) and the category definition (number of categories and exclusiveness of them). It would be interesting to consider automatic detection of the best values of these thresholds.

4.5.2 Possibility of further Improvement

In this section, we discuss possible further improvements for our techniques. We found the following causes of wrong categorization through the experiments above.

[1] *Category definition*

For some Web pages it is difficult to assign only one category out of the 15 Infoseek categories, because some of the categories share topics. For example, since the "computer" category and the "internet" category overlap, some computer-related pages, such as the Stanford Digital Library Project homepage (http://www-diglib.stanford.edu/) should be assigned to both categories. Furthermore, some Web pages refer to multiple topics and should be assigned to quite independent categories. Since our system is able to assign multiple categories to each page according to the "confidence value" based on distribution of similarity values on categories, we should evaluate the accuracy from a viewpoint of multiple categorization.

[2] *Collection of training data set*

In our experiments, we used all of the collected texts as training data. However, some pages might have short texts and some might have extremely long texts. Some might be written in English, but some in other languages. Some might have only index information. We should consider how to collect appropriate training texts. Our system could generate the KB with higher quality by using more appropriate training texts.

[3] *HTML tags*

We used only word frequency to assign keyword weights. However, frequency is not effective for short pages. We believe that HTML tags denoting a title or hyperlinks could be helpful clues to extract keywords and assign their weights more correctly. We may also be able to improve the accuracy not only by using keywords in the page, but also by using URLs which refer to the page and URLs which the page refers to (Google 1997).

[4] *Disambiguation of word meaning*

Since we do not apply any methods for word disambiguation, words with different meaning are regarded as the same if they have the same spelling. Researchers have been working on word disambiguation using co-occurrency of the words (Hinrich 1997, etc.). We should use this kind of technique to distinguish the words according to their meaning and the contexts in which the words are used.

As future research, it would be interesting to consider how to collect appropriate training data semi-automatically from a large set of texts such as the Web and how to update KBs dynamically.

5. Conclusions

In this paper, we presented methods for Web page categorization. The results of our experiments are encouraging. We believe that the IR systems with information organization functions such as categorization enable us to search for information more effectively and comfortably.

Acknowledgment

The authors have benefited from discussions with Stanford's Digital Libraries Project members. This work was supported by NSF, DARPA and NASA under Stanford's Digital Libraries contract.

References

[1]Baldonado, M. Q. W., and Winograd, T. 1997. Sensemaker: An information-exploration interface supporting the contextual evolution of a user's interests. In Proceedings of CHI97, 11-18.

[2]Google. 1997. http://backrub.stanford.edu/.

[3]Hayes, J., and Weinstein, S. P. 1990. CONSTRUE /TIS: A system for content-based indexing of a database of news stories. In Proceedings of Second Annual Conference on Innovative Applications of Artificial Intelligence, 1-5.

[4]Infoseek 1995. Internet directory and query service. http://www.infoseek.com.

[5]Infoseek 1996. Aptex categorizes more than 700,000 web sites for infoseek. http://info.infoseek.com/doc/PressReleases/hnc.html.

[6]Iwayama, M. 1995. Cluster-Based Text Categorization: A comparison of Category Search Strategies. In Proceedings of SIGIR1995, 273-280.

[7]Mase, H.; Tsuji, H,; Kinukawa, H.; Hosoya, Y.; Koutani, K.; and Kiyota, K. 1996. Experimental simulation for automatic patent categorization. In Proceedings of Advances in Production Management Systems, 377-382.

[8]Porter, M. F. 1980. An algorithm for suffix stripping, 130-137.

[9]Python 1998. http://www.python.org/.

[10]Sahami, M.; Yusufali, S.; and Baldonado, M. Q. W. 1998. SONIA: A Service for Organizing Networked Information Autonomously. In Proceedings of the Third ACM Conference on Digital Libraries.

[11]Schutze, H. 1997. Ambiguity Resolution in Language Learning, CSLI Lecture Notes Number 71, CSLI Publications.

[12]Yahoo! 1995. On-line guide for the internet. http://www.yahoo.com/.