CS3243 Assignment 3: Text Categorization

U077151L Lai Xiaoni
U077194E Qiao Li
U077181Y Zeng Qiang

1 kNN Classification Method

1.1 Techniques and design

The basic idea of kNN classification is that the category of a given document is determined by the categories of its k nearest neighbors. For example, if k = 5, the nearest 5 neighbors of a document contains 3 from category A, 1 from category B and 1 from category C, this document is classified as in category A.

It is unnecessary to have a training stage for kNN because any document can be compared with all the documents in the training corpus and subsequently find out its k nearest neighbors to determine its category. However, if among the 5 nearest neighbors, 2 are from category A and 2 are from category B, it is difficult to determine which category this document is in. Also, it is possible that one document is in fact in more than one category. In order to break up the ties as well as identify multi-categories, we need to find out the threshold for being in one category according to the training corpus.

Therefore, the training corpus is divided into two parts. 20% of the training corpus is used as a validation set.

Assume the threshold for category A is θ_A,

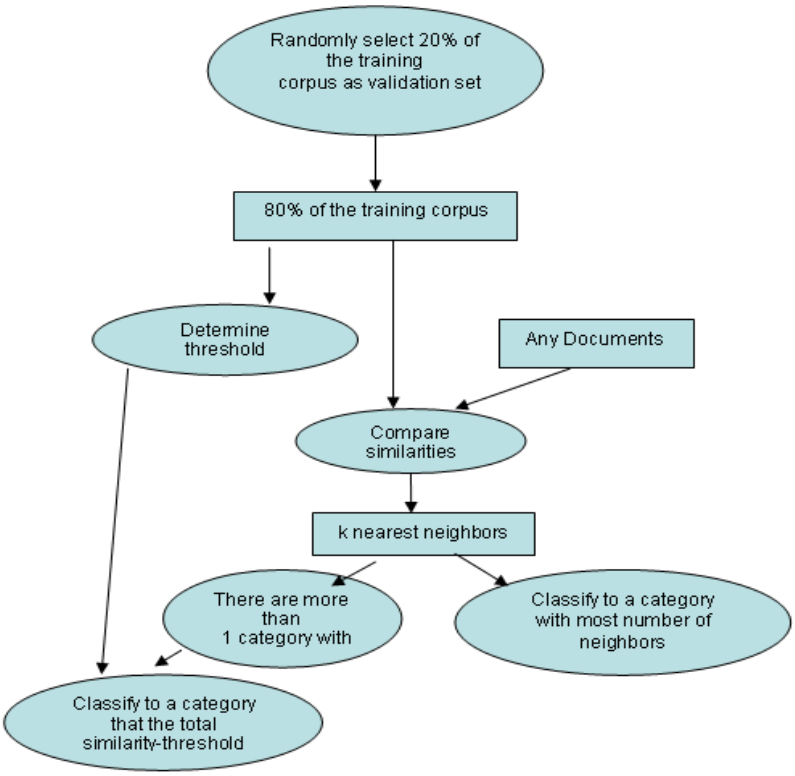
$$\text{document } D \in A \text{ if similarity}(D, A) > \theta_A$$

$$\text{Similarity } (D,A) = \text{similarity}(D, D_1)+\text{similarity}(D, D_2)\ldots+\text{similarity } (D, D_N)$$

$$\text{where } D_1, D_2 .. D_N \in A \text{ and } N < k$$

As it is given that document Q in the validation set $\in$A, similarity(Q, A) >=θ_A

Hence, we can use the validation set to estimate the value of threshold for each category. The threshold for one category has to be smaller or equal to the smallest similarity(Q,A) given Q $\in$ A.



1 Workflow of kNN Classification

1.2 Structure of program

The classes of the program are described as follows:

Doc – contain the path of each document and its category.

DocSimi – contains a Doc object and the similarity. This indicates the similarity between two Docs

Similarity – calculate the similarity between two documents given their paths. This code is given.

kNN – contain the training set and the validation set, value k, the names of the categories and their threshold. The validation process is performed when a new kNN object is created

Evaluator – calculate the precision and output the results of the classification

PerformKNN – perform kNN to determine the categories of a set of documents given k and the main directory of training corpus and the documents.

The main class performing kNN is the kNN class, the main functions are described as follows:

setTrainingSet- set up a list of Docs with the corresponding path of the documents and their categories, which is determined by the name of the directories.

getTopK – given a path of a document, return an array of k DocSimi objects which indicate the similarities between the documents. The similarity is computed as the number of the same terms between two documents. The documents in the training set are sorted according to the similarity to this document.

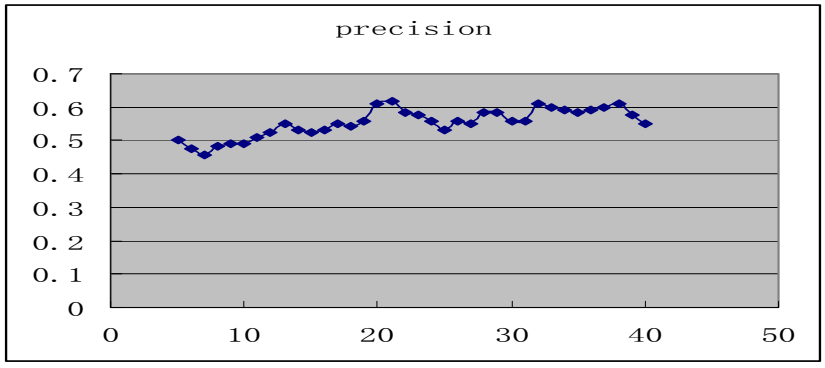
validate – use the validation set to set up the threshold. First, each document in the validation obtain its k nearest neighbors using **getTopK** and get the sum of the similarities of the documents from is own category in the top k. The smallest threshold for each category is chosen.

getCategory – given a path of a document, return the category of a document.

getResult – given a path of a directory of the correct answers, return a list of Docs with its correct category

With the list of correct results and the list of categorization done by kNN classifier, **Evaluator** class is able to count the number of correct classification and determine the precision

1.3 Comparison/analysis of results



2 kNN Evaluation Results

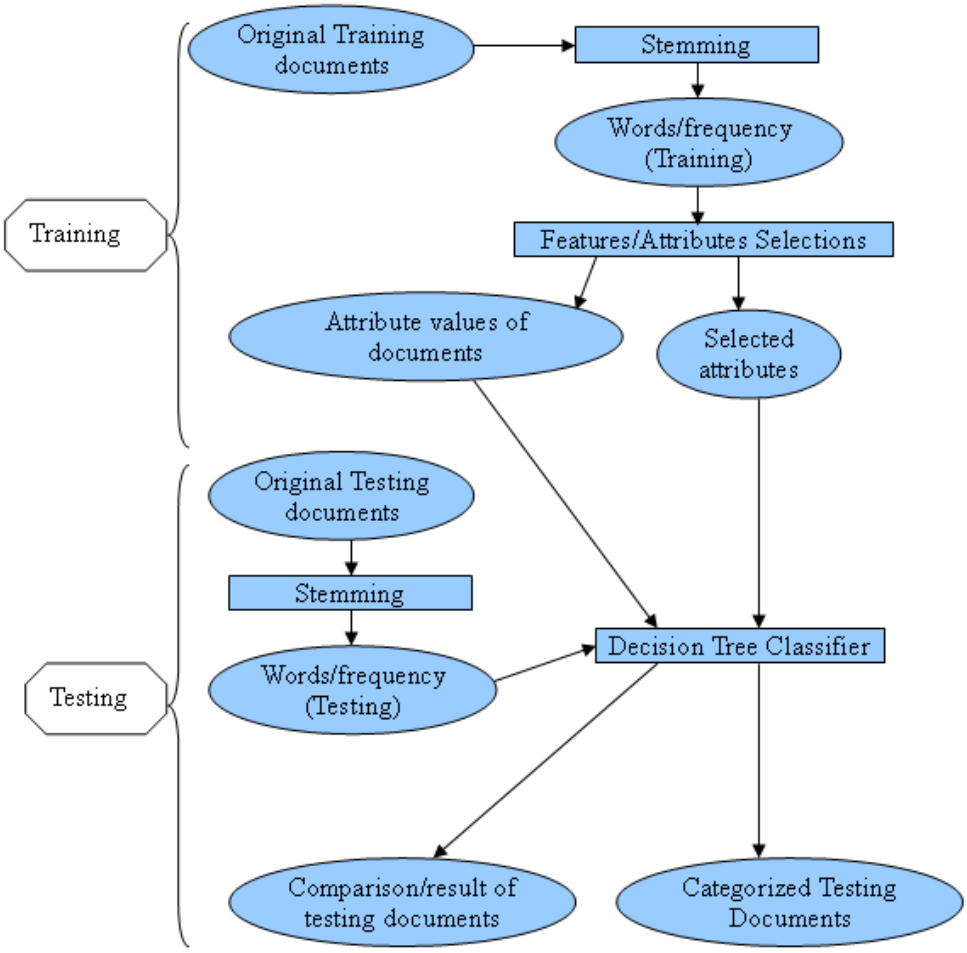
After testing from k = 5 to k = 40, the general precision is between 50% and 60% for kNN classifier. It performs the best when k = 21. However, it performs worse than Decision Tree, which will be described in the next session.

2 Decision Tree Classification Method

2.1 Techniques and Design

2.1.1 Design Overview

The text categorization process is divided into two stages: training stage and testing stage. The following diagram illustrates the work-flow of the decision tree text categorization in this assignment.



3 Decision Tree classification workflow

In this diagram, “stemming” refers to the some pre-processing on text documents. This includes converting words to lower case, removing stop words and reduce words to “morph” form by removing suffixes such as -SES, -ING, etc….

In this assignment, this stemming process has been implemented by the given executable file “stemmer.jar”. We input all original training and testing documents into stemmer.jar and the corresponding files would be generated. These output files retain the most important feature from the original document: morph-form word and its occurrence in that document.

The “Feature Selection” process is implemented by an algorithm written by us. This algorithm takes in the abovementioned output files, which are lists of word and its frequency, and analyze to generate a set of attributes useful for producing a decision tree. Later, files generated by Feature Selection algorithm would be fed into C5.0, a decision tree package freely available from Web, to generate the desired results.

Both Feature selection process and Decision Tree classification process would be explained in the following sections.

2.1.2 Feature Selection

The Feature Selection algorithm basically works in three phases:

1. Generating attributes
2. Assigning attribute values to files in training set
3. Assigning attribute values to files in testing set.

In the beginning, all documents in each category would be analyzed. Those words which occur with very low frequency in each document would be removed, because normally their existence does not imply any characteristic of the category to which this document belongs; this frequency threshold is called “local threshold”.

Later, we calculate the document frequency (df) of each word per category; this is done by computing how many documents in one category a word has appear. Those words which have a document frequency high enough are considered to be attributes; this is because words appearing in multiple documents in one category are normally characteristic words in that category. We set a document frequency threshold in order to judge how high the df a word must possess in order to be retained after feature selection. We call this threshold as “global df threshold”.

During implementation, we realized that the number of documents in different categories vary from 12 to 21. This means that the global df threshold cannot be the same from category to category; otherwise this is “unfair” to those words in categories with fewer documents because they must appear quite frequently in order to be considered as attributes. Therefore, we set a constant “category df threshold percentage” which acts as a fixed percentage of “global df threshold” with respect to number of documents in one category. This solves the problem.

Now, a set of attributes has been produced. The Feature Selection algorithm would then loop through all the documents in training set and testing set, and write to a file indicating whether these attribute words have appeared inside the documents. The generated files are in the format required by the C5.0 decision tree package.

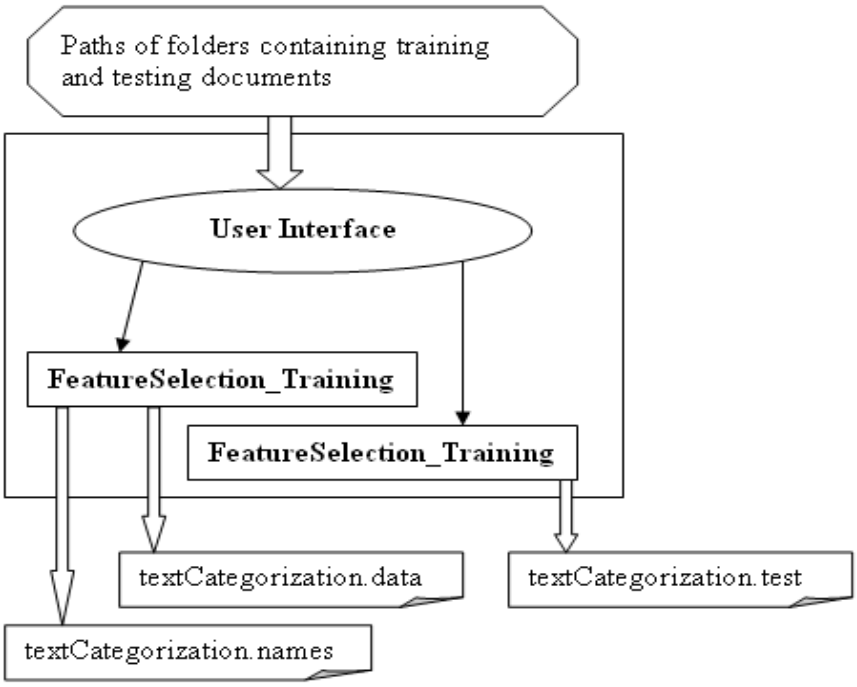
2.1.3 Decision Tree Classifier

We have adopted C5.0 package available from Web for our own purpose. This package requires users to feed in a list of attributes the feature selection algorithm has chosen, two data files containing attribute values of documents in training set and testing set respectively.

After inputting the three files we have generated during feature selection stage, the C5.0 would automatically analyze the testing documents and categorize them. The decision tree generated would be written to a new file; and the categorization results produced by classifier would be compared with expert results (the actual categories of these testing documents).

2.2 Program Structure

The Feature Selection algorithm is implemented with a simple UI class and two other classes which deal with training set and testing respectively. The following diagram illustrates the program structure of the Feature Selection algorithm.



4 Work-flow of Feature Selection Implementation

The three generated files (textCategorization.data, textCategorization.test, textCategorization.names) would be fed into C5.0 for the final results.

2.3 Evaluation of results

After many times of execution, we realized that the best "local threshold" (see 2.1.2) is 2. When words with frequency less than 2 in local documents are removed, the decision tree classifier can achieve the best results.

The next task is to determine the best "category df threshold percentage" (see 2.1.2). We have created classifier using different percentage from 0.1 to 0.5. The testing results with different pruning setting are presented below.

Note: error = 1-precision; threshold = category df threshold percentage.

threshold 0.1			
	Pruning CF %	tree size	error(%)
	0	33	51.7
	25	30	50.8
	50	35	51.7
	75	35	51.7
average			51.475

threshold 0.2			
	Pruning CF %	tree size	error(%)
	0	13	9.2
	25	11	10
	50	11	10
	75	11	10
average			9.8

threshold 0.3			
	Pruning CF %	tree size	error(%)
	0	13	9.2
	25	11	10
	50	11	10
	75	11	10
average			9.8

threshold 0.4			
	Pruning CF %	tree size	error(%)
	0	15	25
	25	13	25.8
	50	13	25.8
	75	15	25
average			25.4

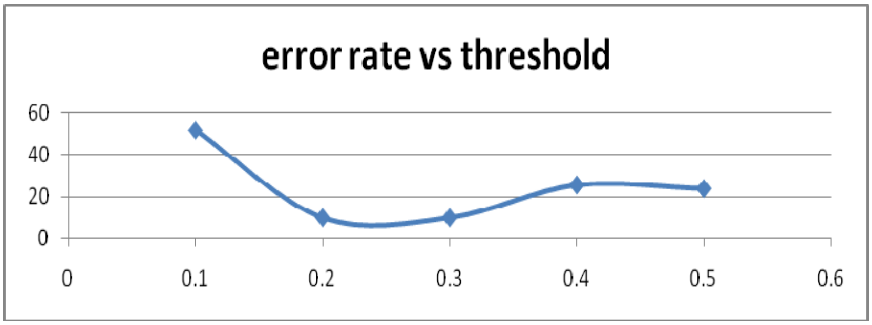
threshold 0.5			
	Pruning CF %	tree size	error(%)
	0	18	25
	25	16	23.3
	50	16	23.3
	75	16	23.3
average			23.725

threshold	error
0.1	51.5
0.2	9.8
0.3	9.8
0.4	25.4
0.5	23.7

5 Decision Tree Results Evaluation

The existence of pruning CF can reduce branches. However, due to the small size of dataset, the effect of pruning is very limited. And as the pruning factor go towards 1, less and less branches are pruned. In fact, we produce better result without pruning.

The last figure summarizes all five tables with threshold values and average error rate of different pruning settings. The table is also drawn to graph.



6 Error rate vs. Threshold

From the graph we find that, the decision tree produces lowest error rate when the percentage is between 0.2 and 0.3 with an error rate lower than 10%.

Thus we conclude that by using global threshold percentage of 0.25 we can construct a text classified with a precision of less than 10%