KAN and RinSCut: Lazy Linear Classifier and Rank-in-Score Threshold in Similarity-Based Text Categorization

Kang Hyuk Lee KANGL@IT.USYD.EDU.AU
Judy Kay JUDY@IT.USYD.EDU.AU
School of Information Technologies, University of Sydney, NSW, 2006, Australia

Byeong Ho Kang BHKANG@UTAS.EDU.AU
School of Computing, University of Tasmania, Hobart, Tasmania, 7001, Australia

Abstract

Two important research areas in statistical approaches for automated text categorization are similarity-based learning algorithms and associated thresholding strategies. The combination of these techniques significantly influences the overall performance of text categorization systems. After researching common techniques in both areas, we describe a lazy linear classifier known as the keyword association network (KAN) and a rank-in-score (RinSCut) thresholding strategy to improve the categorization performance over existing techniques. Extensive experiments have been conducted on the Reuters-21578 data set. The experimental results show that KAN outperforms two linear classifiers, Rocchio and Widrow-Hoff, and gives results competitive with k-NN. All implemented classifiers except Widrow-Hoff show performance improvements with RinSCut.

1. Introduction

An important approach to management of large amounts of textual information is to classify documents into topics. Traditionally, experts who are knowledgeable about the topics perform text categorization manually. This requires substantial human resources. Given that the amount of online textual information is growing rapidly, the need for reliable automatic text categorization has been increasing.

There has been a wide range of statistics-based learning algorithms applied to this automatic text categorization task. These include the linear classifiers [Joachims, 1997; Lewis et al., 1996; Buckley, Salton, & Allan, 1994], k-Nearest Neighbor (k-NN) classification [Yavuz & Guvenir, 1998; Yang, 1994], naive Bayes probabilistic classification [Joachims, 1997; Lewis & Ringuette, 1994; McCallum & Nigam, 1998], support vector machines [Joachims, 1998], and neural networks [Wiener, Pedersen, & Weigend, 1995]. The main task of a classifier is to give more weight to the informative words than it assigns to non-informative ones. A general group of classifiers that has been shown good categorization performance is similarity-based. These classifiers assign relevance similarity scores to every document-category pair. These classifiers can be grouped into two classes: profile-based linear learning algorithms and instance-based lazy classifiers. The first class contains Rocchio [Rocchio, 1971] and Widrow-Hoff [Widrow & Stearns, 1985] linear algorithms that prepare a generalized profile set for each category in which each feature (i.e., words) has a weight vector computed from a set of training documents. The k-NN classifier can be considered a lazy learning algorithm because it does not involve the pre-learning process that determines a generalized absolute weight for each feature. An instance-based classifier uses training examples *directly* in the similarity computation, and its motivation is based on the reasoning that a document itself has more representative power than a generalized category feature vector.

An important challenge in similarity-based text categorization is to exploit co-location information. Intuitively, it seems likely that we should be able to achieve better predictions for document similarity if we can take account of pairs of words that occur relatively frequently in a document. Such co-location information has been used in semantic feature indexing [Deerwester, Dumais, & Harshman, 1990], automatic thesaurus generation [Crouch & Yang, 1992], and various text mining tasks. While extracting this information from a large document set has the potential to be computationally expensive, its appropriate use has several appealing properties when learning a classifier.

Another critical area in assessing document similarity is the discriminating power function used to identify informative features. In [Mladenic & Grobelnik, 1999], a number of algorithms were investigated and applied to feature selection as a preprocessing step. This stage selects a subset of features. Using the discriminating power

(numeric value) of each feature in similarity computation is a promising approach in similarity-based text categorization [Shankar & Karipis, 2002].

The mapping from a new document to relevant categories is achieved by thresholding the similarity scores computed in a classifier. Existing common thresholding techniques for online categorization are rank-based thresholding (RCut) and score-based optimization (SCut). These techniques have been extensively evaluated on various corpora in [Yang, 1999; Yang, 2001]. RCut finds a rank threshold that optimizes the global performance and applies this to all the categories. As a result, this strategy will give low performance in the rare categories which have small number of documents. By contrast, SCut learns a similarity score for each category so as to optimize the local category performance. But, it does not guarantee global performance. Finding the optimal thresholding strategy for any given classifier and data set is difficult and combining the strengths of the existing strategies for overcoming this difficulty is a challenge in text categorization [Yang, 2001].

In this paper, we present a lazy linear classifier - KAN, and a new thresholding method - RinSCut that is designed to combine the best features of RCut and SCut. Like the lazy instance-based algorithms, KAN computes the feature weights when a new document is available. However, it uses generalized co-location information prepared from the training documents rather than using the instances directly. The empirical results on the Reuters-21578 show that KAN outperforms two representative linear classifiers, Rocchio and Widrow-Hoff, and produces slightly higher performance than k-NN. RinSCut with KAN, Rocchio, and k-NN shows significant performance improvements.

The rest of this paper is organized as follows. Section 2 reviews the common techniques in similarity-based text categorization. An introduction to the process of building KAN and applying it to a text categorization task is described in Section 3. Section 4 explains our new thresholding strategy, RinSCut. Experimental setup and results are given in Section 5, followed by the concluding remarks in Section 6.

2. Similarity-Based Text Categorization

2.1 Representation

The common representation adopted by most learning algorithms is the "bag-of-words" representation. In this representation, each document d is transformed to a vector $v = (x_1, x_2, \ldots, x_n)$. Here, n is the number of unique features and each x_i is the weighting value of the ith feature. This is calculated as a combination of two common weighting schemes: the term frequency, TF_i, is the number of times the ith feature occurs in document d and the inverse document frequency, IDF_i, is $log(|N|/DF_i)$, where DF_i is the number of documents that contain the ith feature and $|N|$ is the total number of documents in the training set. Then, x_i is $TF_i \times IDF_i$. Because the document lengths may vary widely, a length normalization factor is applied to the term weighting function. The weighting equation that is used [Buckley et al., 1995] in this experiment is given as:

$$x_i = \frac{(logTF_i + 1.0) \times IDF_i}{\sqrt{\sum_{i=1,n} [(logTF_i + 1.0) \times IDF_i]^2}}$$

2.2 Similarity-based classifiers

2.2.1 PROFILE-BASED LINEAR ALGORITHMS

This class of linear algorithms attempts to learn a generalized profile in the form of a weight vector for each category. The typical linear learning algorithm is based on Rocchio relevance feedback [Rocchio, 1971]. In this algorithm, each category c has a vector of the form $w = (y_1, y_2, \ldots, y_n)$ and each y_i in this prototype vector w is:

$$y_i = [\alpha \times \sum_{d \in c} x_i(d) \times |C|^{-1}] - [\beta \times \sum_{d \notin c} x_i(d) \times (|N| - |C|)^{-1}]$$

where α and β are adjustment parameters for positive and negative examples, $|C|$ is the number of positive documents in the category C, and $|N|$ is the total number of documents in the training set. The similarity value between a category and a new document is obtained as the inner product between the two corresponding feature vectors. The problem in this classifier is that some informative features in a rare category will have small weights if they appear occasionally in the negative examples. One of the recently proposed linear classifiers is Widrow-Hoff (WH) [Widrow & Stearns, 1985]. WH is an on-line classifier that updates weight vectors by using one training example at a time. For each category, the new weight of ith feature, $y_{i,j+1}$, is calculated from the old weight vector, $\mathbf{w}_j$, and new document vector, $\mathbf{v}_j$:

$$y_{i,j+1} = y_{i,j} - 2\eta(\mathbf{w}_j \cdot \mathbf{v}_j - b_j)x_{i,j}$$

where $\mathbf{w}_j \cdot \mathbf{v}_j$ is the cosine value of the two vectors, η is the learning rate parameter, and b_j is the category label of new document having 1 if the new document is positive and 0 if it is negative. In previous evaluation [Lewis et al., 1996], WH has shown the improved performance over Rocchio.

2.2.2 INSTANCE-BASED LAZY ALGORITHM

The k-Nearest Neighbor (k-NN) classifier is an instance-based learning algorithm that operates directly on training documents without generating category profiles. The essential idea behind this algorithm is that a document itself has more representative power than the generalized category feature vector. For a new document, k-NN computes its similarity scores to all the training documents. These similarity scores are then sorted into descending order. The final score for a category is the sum of the similarity scores of the documents in that category using the k top-ranking documents. While giving good performance, k-

NN has several drawbacks. In k-NN, it is difficult to find the optimal k while there are uneven distributions of training examples across categories. Also, due to the lack of generalized feature weights, some noisy examples can have a significant impact on the quality of the ranking. Furthermore, the time taken for the similarity calculation increases with the size of the training data set.

2.3 Thresholding strategy

For the similarity-based classifiers, a thresholding strategy is critical. This subsection describes two common techniques: rank-based and score-based.

2.3.1 RANK-BASED THRESHOLD

The rank-based threshold (RCut) sorts the similarity scores of categories for each document. Then, it assigns a "YES" decision to the t top-ranking categories. This threshold, t, is predefined automatically. This is set to optimize the global performance in the training set by assuming that all the test documents belong to the same number of categories. It usually gives good micro-averaged performance since the selection of t value mainly depends on the large categories. However, when this globally optimized threshold t is 1 and many rare categories have overlapping concepts with other categories, this strategy may result in low macro-averaged performance.

2.3.2 SCORE-BASED THRESHOLD

The score-based strategy (SCut) learns the optimal threshold for each category. The optimal threshold is the similarity score that optimizes the performance measure of each category. If the local performance of each category is the primary concern and the test documents belong to a variable number of categories, this strategy may be a better choice than RCut. However, it is not trivial to find an optimal threshold for each category. This problem becomes more apparent with small sets of training data.

3. Keyword Association Network: A Lazy Linear Learning Algorithm

In this section, we describe a new lazy linear learning technique called the keyword association network (KAN). The main motivation for KAN is to achieve the following two important objectives: (1) to give a feature the appropriate weight according to its semantic meaning and importance in a given document and (2) to remove the influence of irrelevant features by giving more weight to the discriminating features. In the following subsections, we present KAN and explain the process of applying it to text categorization.

3.1 Keyword association network (KAN)

Previous work showed that it is possible to automatically find words that are semantically similar to a given word based on the co-location of words [Ruge, 1992; Sekine et al., 1992]. Our goal was to use this type of statistical information to determine the importance and the semantic meaning of a word in a given document. KAN is constructed by means of a network representation based on co-location information. The degree of relationship between two words is represented by a confidence value. This measure was used in finding association rules in [Agrawal et al., 1996] that have been identified as an important tool for knowledge discovery in huge transactional databases. In KAN, the confidence value is used for measuring how the presence of one word in a given document may influence the presence of another. When the category c has a set of n unique features $\{F = (w_1, w_2, \ldots, w_n)\}$, the ith feature's confidence value to the jth feature, $CONF(w_i,w_j)$, is:

$$CONF(w_i,w_j) = \frac{DF(w_i,w_j \mid c)}{DF(w_i \mid c)}$$

where $DF(w_i,w_j \mid c)$ is the document frequency of the two co-locating features w_i and w_j in the category c and $DF(w_i \mid c)$ is the document frequency of the feature w_i in the category c. Note that this rule is asymmetric, i.e. $CONF(w_j,w_i)$ has a different value from $CONF(w_i,w_j)$ due to the different denominator. Unlike the profile-based linear classifiers, KAN does not fix the feature weights through a pre-learning process. The absolute weight for each feature in a certain category is determined by seeing a new document. This is why we call KAN a lazy learning algorithm.

When a new document g having the vector of the form $(z_1, z_2, \ldots, z_n)$ is available, the weight of ith feature in a category c having the weight vector $(y_1, y_2, \ldots, y_n)$ is computed as follows:

$$y_i = [\sum_{d \in c} x_i(d) \times |C|^{-1}] \times [1 + \sum_{j \neq i} (CONF(w_j,w_i) \times l_j)]$$

where $x_i(d)$ is the weight of ith feature in a training example d, l_j is the existence label of jth feature in a new document having 1 if the feature appears in the new document and 0 if not. Then, the similarity value between the category c and a new document g is the dot product between two vectors.

$$\text{Similarity}(c, g) = \sum_{i=1,n} (y_i \times z_i)$$

A potential drawback of KAN is that it would be impractical with a high dimensional feature space. When calculating the weight of ith feature in a category, KAN requires a confidence value for each pair of words. The computational complexity of KAN is $O(n^2)$ in both constructing time and similarity calculating time where n is the number of unique words in the training set. So, it is critical to reduce the feature space by applying feature selection algorithms. In our experiments, we have applied information gain measures in feature selection and then chosen 50 words for each category.

3.2 Discriminating power function

An important factor in achieving high performance in text categorization is to reduce the influence of the large number of irrelevant features that occur evenly across the categories. In KAN, we apply a discriminating power function similar to the scheme used in [Shankar & Karipis, 2000].

Let m be the number of categories. For the ith feature, f_i, we calculate a weight vector $h_i = (df_{1,i}, df_{2,i}, \ldots, df_{m,i})$. Each $df_{j,i}$ is $DF(f_i|c_j) / DF(f_i)$ where $DF(f_i|c_j)$ is the document frequency of f_i in the jth category and $DF(f_i)$ is the document frequency of f_i in the training set. Shankar computed the discriminating power value of the ith feature, P_i, as follows:

$$P_i = \sum_{j=1,m}(\hat{h}_{j,i})^2$$

where $\hat{h}_i$ is one-norm scaled vector of h_i (so, the sum of all the components in $\hat{h}_i$ is to be 1). This value will have the lowest value, $1/m$, if a feature is evenly distributed across all the categories and the largest value, 1, if it appears in only one category. One drawback of this scheme is that it causes a steeply declining tendency in the weights of some informative features in the several categories that have the same overlapping concept. Note that we mentioned this problem in the Rocchio classifier. In KAN, we use $df_{j,i}$ as the discriminating power of the ith feature in the jth category. To reduce the impacts of the irrelevant features, $df_{j,i}$ is transformed to $S_{j,i}$ as follows:

$$S_{j,i} = df_{j,i}^{(\lambda / df_{j,i})}$$

where the range of λ is $0 < \lambda < 1$. The graph of the df and S values of the features has an S shape at the point of λ. This discriminating power value is then used in the computation of the category weight vector mentioned in Section 3.1 as follows:

$$y_i = [\sum_{d \in c} x_i(d) \times |C|^{-1}] \times [1 + \sum_{j \neq i}(CONF(w_j,w_i) \times S_i \times l_j)]$$

4. Rank-in-Score Based Threshold

The properties of the classifier, the choice of an evaluation method, and the characteristics of data set all exercise important effects on the selection of the thresholding strategy. In the absence of an outstanding strategy, it is not trivial to find the optimal method. Addressing this problem is a challenge in similarity-based text categorization. One way of overcoming the above problem is to design a new thresholding strategy that jointly uses the strengths of different strategies.

With the multi-class categorization problem in which documents belong to a variable number of categories, the score-based threshold (SCut) seems to be optimal because the rank-based threshold (RCut) recommends the same number of categories to all new documents. The SCut finds a different similarity score for each category and so optimizes local performance while RCut gives one single rank to all the categories by optimizing the global performance.

We define the rank-in-score threshold (RinSCut) to jointly use the strengths of two strategies and, as a result, to overcome weaknesses in both SCut and RCut. The RinSCut finds two threshold scores, s_{top} and s_{bottom}, for each category. When $\boldsymbol{s}$ is the optimal score from SCut, let R be the set of negative documents with similarity scores above $\boldsymbol{s}$ and T be the set of positive documents with similarity scores below $\boldsymbol{s}$. Then, two threshold scores are defined as follows:

$$s_{top} = \boldsymbol{s} + \frac{\sum_{i \in R}(v_i - \boldsymbol{s})}{|R|}$$

$$s_{bottom} = \boldsymbol{s} - \frac{\sum_{i \in T}(\boldsymbol{s} - v_i)}{|T|}$$

where v is similarity value of document, $|R|$ and $|T|$ are the number of documents in R and T respectively. The range of similarity values between these two threshold scores is considered as the ambiguous zone: the categorization decision for a test document to this category can not be made only with the similarity score. For the test documents that belong to this zone, the rank threshold is used to make the final decision.

For an assignment decision on a test document d to a category c with the similarity score, Similarity(d, c), assigns a "YES" decision if Similarity(d, c) $\geq s_{top}$ and a "NO" decision if Similarity(d, c) $< s_{bottom}$. If Similarity(d, c) is between s_{top} and s_{bottom}, the assignment decision depends on the rank threshold t. The t threshold can be defined by two ways, one optimizing the global performance (GRinSCut) and the other optimizing the local performance of each category (LRinSCut).

5. Experiments

5.1 Data set

The data set is the Reuters-21578 text categorization test collection. In this collection, articles contain no meta-

Round	1	2	3	4	5	6	7	8	9	10
No. of unique data	106	212	371	689	1,272	2,409	3,696	5,136	6,202	6,984

Table 1. Number of unique training data in each round.

information (hyperlinks or tags) that could be used for extracting features. We split the articles into training and test sets according to the Modified Lewis Split. Instead of analyzing all 135 categories, we choose the categories with at least 10 articles in both training and test sets. The number of selected categories is 53. This results in a corpus of 6,984 in the training data and 3,265 in the test data. In the training and test data sets, many articles belong to multiple categories.

5.2 Experimental setup

For the preprocessing steps, we applied a stop-list and Porter's stemming algorithm [Porter, 1980] to the articles. We used information gain for feature selection and took the same number of features, 50 in these experiments, for all categories. We have implemented four classifiers (Rocchio, WH, k-NN, and KAN) with three thresholding strategies (SCut, GRinSCut, LRinSCut). The performance was measured using the standard F_1 measure that is designed to balance recall and precision by giving them equal weight [Yang, 1999].

$$F_1 = 2 \times \text{Recall} \times \text{Precision} / (\text{Recall} + \text{Precision})$$

We computed the macro-averaged F_1 as well as the micro-averaged F_1 in order to analyze the performance on individual categories. We conducted the experiments by increasing the size of the training data to construct learning curves for the classifiers. Each experiment was repeated 10-fold. Table 1 shows the amount of training data in each round. The training set for each round is a superset of the one for the previous round. For KAN, we used $\lambda = 0.4$ for the discriminating power values. To compute the vectors of categories, we used $\alpha = 16$ and $\beta = 4$ for the Rocchio as suggested in [Buckley, Salton, & Allan, 1994] and $\eta = 0.25$ for the WH. The value k used in these experiments for the k-NN is 10, 20, 30, 40, 50, and 100. Then, we chose the value with the best result in each round. Table 2 presents the k value of k-NN that gives the best performance for each round.

Round \ Threshold	SCut	GRinSCut	LRinSCut
1	20	20	20
2	40	50	30
3	40	50	50
4	10	20	30
5	10	50	20
6	10	30	50
7	10	40	50
8	10	10	40
9	10	10	40
10	10	20	40

Table 2. k values of k-NN classifier that gives the best performance for each round in each thresholding strategy.

5.3 Results

Category \ Classfier	Rocchio	WH	k-NN	KAN
earn	0.961	0.956	**0.966**	0.955
acq	0.563	0.726	0.491	**0.735**
money-fx	0.361	0.525	**0.679**	0.630
grain	0.428	**0.752**	0.720	0.723
crude	0.617	**0.715**	0.319	0.704
trade	0.369	0.575	**0.653**	0.586
interest	0.307	0.411	0.447	**0.556**
wheat	0.370	**0.737**	0.418	0.727
ship	0.762	**0.792**	0.782	0.762
corn	0.257	**0.630**	0.524	0.624
all macro avg.	0.412	0.570	0.570	**0.584**
all micro avg.	0.634	0.681	0.692	**0.750**

Table 3. F_1 performance of the ten most frequent categories and macro and micro-averaged F_1 with SCut.

Category \ Classifier	Rocchio	WH	k-NN	KAN
earn	0.975	0.961	**0.977**	0.965
acq	0.819	0.826	**0.937**	0.860
money-fx	0.414	0.689	**0.768**	0.668
grain	0.265	0.729	**0.775**	0.558
crude	0.715	0.779	0.739	**0.811**
trade	0.646	0.671	**0.709**	0.646
interest	0.597	0.638	**0.651**	0.540
wheat	0.647	0.450	0.294	**0.762**
ship	**0.819**	0.667	0.816	0.802
corn	0.421	0.524	0.462	**0.568**
all macro avg.	0.549	0.517	0.552	**0.615**
all micro avg.	0.736	0.732	**0.790**	0.780

Table 4. F_1 performance of the ten most frequent categories and macro and micro-averaged F_1 with GRinSCut.

Table 3, Table 4 and Table 5 show the F_1 performance of the ten most frequent categories and the micro and macro averaged F_1 performance of all categories with SCut, GRinSCut, and LRinSCut respectively when all the training examples are used. In Table 3, WH gives the best results in 5 categories while k-NN and KAN achieve the best results in 3 and 2 categories respectively. KAN shows the best performance in both micro and macro averaged F_1. In Table 4 and Table 5, Rocchio, k-NN and KAN with the RinSCut strategies achieve a performance improvement over SCut. By contrast, the performance of WH with the LRinSCut strategy becomes much worse than with SCut. This tendency is mainly caused by very low performance in the rare categories. Note the improvement of the micro-averaged performance in k-NN.

This result matches well with the fact that k-NN is a rank-based classifier. KAN with the variants of RinSCut shows competitive results to k-NN in the micro-averaged F_1 and better results than k-NN in the macro-averaged F_1.

Classifier / Category	Rocchio	WH	k-NN	KAN
earn	0.942	0.961	**0.976**	0.965
acq	0.819	0.826	**0.937**	0.860
money-fx	0.578	0.689	**0.732**	0.668
grain	0.630	0.729	**0.767**	0.708
crude	0.777	0.770	0.749	**0.811**
trade	0.646	0.609	**0.731**	0.646
interest	0.597	**0.658**	0.647	0.540
wheat	0.647	0.766	0.705	**0.845**
ship	**0.819**	0.584	0.798	0.691
corn	0.523	**0.654**	0.600	0.632
all macro avg.	0.573	0.442	0.578	**0.629**
all micro avg.	0.752	0.438	**0.786**	**0.786**

Table 5. F_1 performance of the ten most frequent categories and macro and micro-averaged F_1 with LRinSCut.

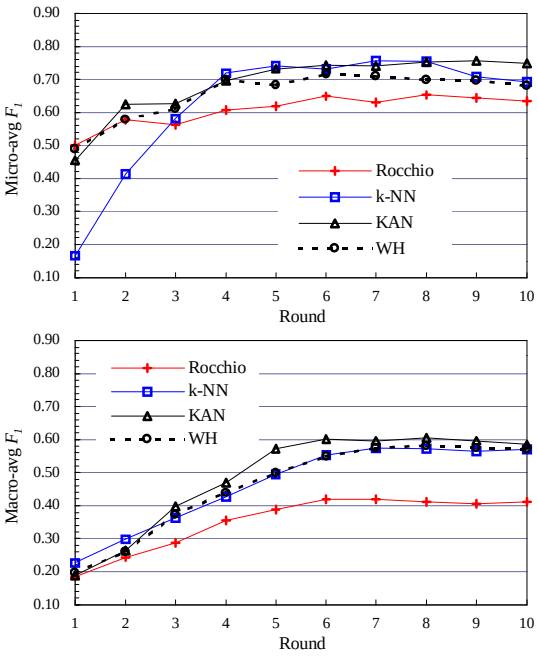

Figure 1. Micro-averaged and macro-averaged F_1 of each classifier with SCut.

Figure 1 depicts the learning curve of each classifier with SCut in the micro and macro-averaged F_1. Rocchio gives lower performance than the other classifiers in both measures across all the rounds with the exception of the lowest micro-averaged performance of k-NN on round 1 and 2. k-NN shows similar performance to KAN after round 3 in the micro-averaged F_1. In macro-averaged F_1, KAN, in general, achieves better results than the other classifiers.

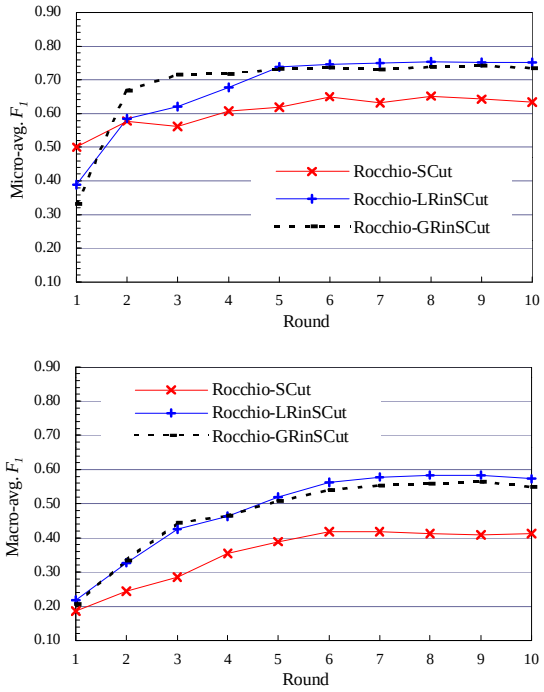

Figure 2. Performance comparison of SCut and RinSCut variants in Rocchio.

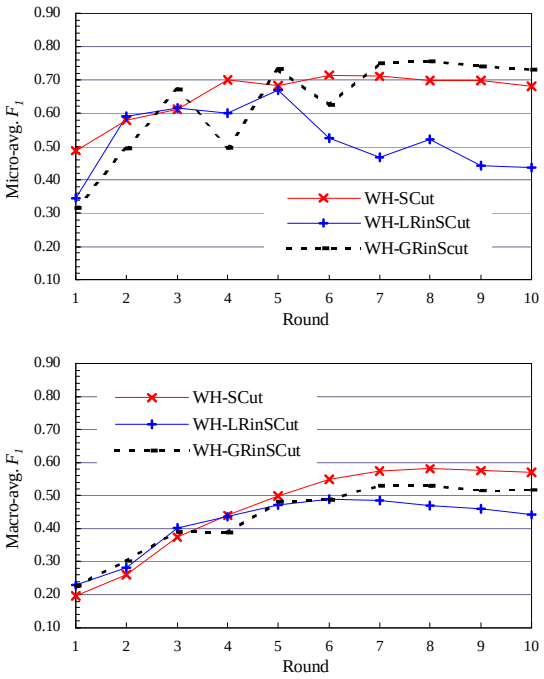

Figure 3. Performance comparison of SCut and RinSCut variants in Widrow-Hoff.

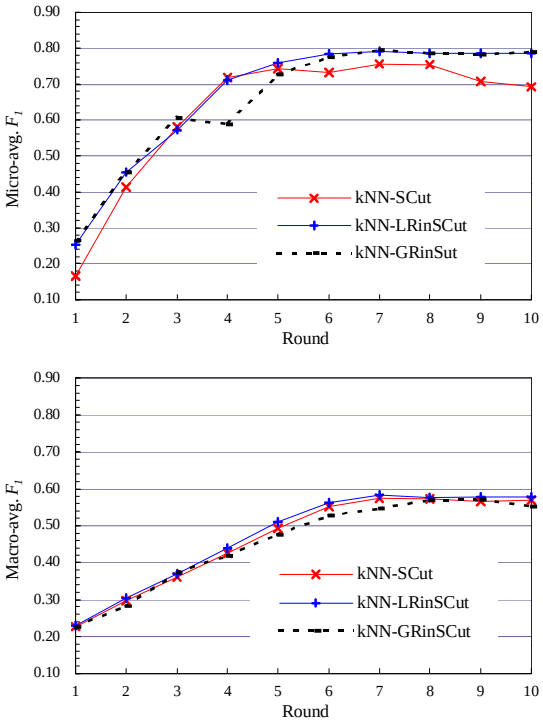

Figure 4. Performance comparison of SCut and RinSCut variants in k-NN.

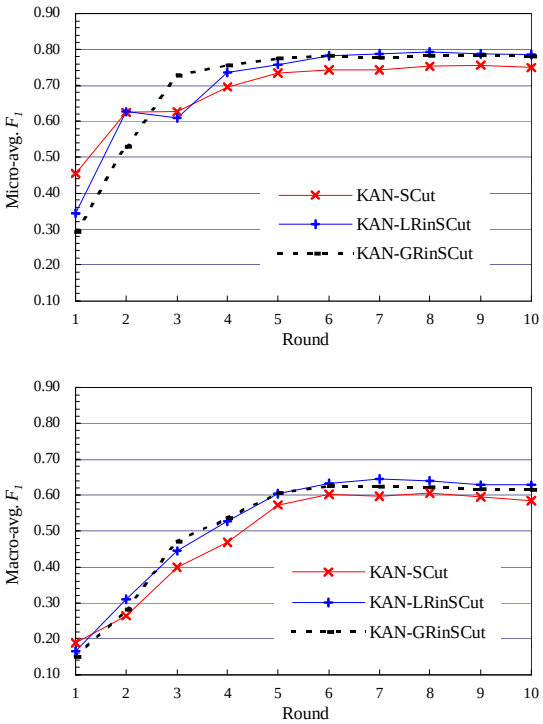

Figure 5. Performance comparison of SCut and RinSCut variants in KAN.

The performance comparison of three thresholding strategies (SCut, GRinSCut, and LRinSCut) in each classifier is shown in Figures 2 - 5. Figure 2 shows that, across all the rounds, RinSCut variants consistently give considerable performance improvements to Rocchio against SCut in the micro and macro-averaged F_1. However, WH with two RinSCut variants, in Figure 3, gives unstable micro-averaged performance and, in general, lower performance than SCut in both measures. In Figure 4, k-NN with RinSCut strategies achieves slightly better micro-averaged performance than SCut while giving similar macro-averaged performance to SCut across all the rounds. Figure 5 shows that KAN with RinSCut strategies achieves slightly better results than SCut in both measures except round 1 and 2 with the small number of training examples.

6. Concluding Remarks

In the statistics-based text categorization, we have explored two main research areas, similarity-based learning algorithms and thresholding strategies. After outlining representative techniques in both areas, we described KAN as a new representation and lazy linear learning algorithm and RinSCut as a new thresholding strategy. We implemented Rocchio, Widrow-Hoff, k-NN, and KAN with SCut, and two RinSCut variants. Extensive experiments have been conducted on the Reuters-21578 collection. Empirical results show that KAN outperforms two linear classifiers, Rocchio and WH, and achieves slightly better results than k-NN. With two variants (GRinSCut and LRinSCut) of RinSCut, all classifiers except WH show considerable performance improvement in micro and macro-averaged F_1. More research is envisaged to investigate the performance of our new approaches in other test data sets.

References

Agrawal, A., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). *Fast Discovery of Association Rules*. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy (Ed.), Advances in Knowledge Discovery and Data Mining (pp. 307-328). AAAI/MIT Press.

Buckley, C., Salton, G., & Allan, J. (1994). *The Effect of Adding Relevance Information in a Relevance Feedback Environment*. International ACM SIGIR Conference (pp. 292-300).

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). *Automatic Query Expansion Using SMART: TREC 3*. The Third Text Retrieval Conference (TREC-3). National Institute of Standards and Technology Special Publication 500-207. Gaithersburg, MD.

Crouch, C. J. & Yang, B. (1992). *Experiments in Automatic Statistical Thesaurus Construction*, SIGIR 92 (pp. 77-88).

Deerwester, S., Dumais, S. T., & Harshman, R. (1990). *Indexing by Latent Semantic Analysis*. Journal of the Society for Information Science, (pp. 391-407), 41, 6.

Joachims, T. (1997). *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*. In Proceedings of the 14th International Conference on Machine Learning ICML'97 (pp. 143-151).

Joachims, T. (1998). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. In European Conference on Machine Learning (ECML-98).

Lewis, D. D. & Ringuette, M. (1994). *Comparison of Two Learning Algorithms for Text Categorization*. In Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94). Nevada, Las Vegas.

Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). *Training Algorithms for Linear Text Classifiers*. In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 298-306).

McCallum, A. & Nigam, K. (1998). *A Comparison of Event Models for Naive Bayes Text Classifiers*. In AAAI-98 Workshop on Learning for Text Categorization.

Mladenic, D. & Grobelnik, M. (1999). *Feature Selection for Unbalanced Class Distribution and Naive Bayes*. In Proceedings of the 16th International Conference on Machine Learning (ICML-99).

Porter, M. F. (1980). *An Algorithm for Suffix Stripping*. Program (pp. 130-137), 14, 3.

Rocchio, J. (1971). *Relevance Feedback in Information Retrieval*. In G. Salton (Ed.), The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall.

Ruge, G. (1992). *Experiments on Linguistically Based Term Associations*. Information Processing & Management (pp. 317-332), 28, 3.

Sekine, S., Carroll, J., Ananiadou, A., & Tsujii, J. (1992). *Automatic Learning for Semantic Collocation*. Proceedings of the Third Conference on Applied Natural Language Processing (pp. 104-110). ACL.

Shankar, S. & Karipis, G. (2000). *A Feature Weight Adjustment for Document Categorization*. The 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 20-23, Boston, MA, USA.

Widrow, B. & Stearns, S. D. (1985). Adaptive Signal Processing. Prentice-Hall Inc., Eaglewood Cliffs, NJ.

Wiener, E., Pedersen, J. O., & Weigend, A. S. (1995). *A Neural Network Approach to Topic Spotting*. In Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95).

Yang, Y. (1994). *Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval*. In Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 13-22).

Yang, Y. (1999). *An Evaluation of Statistical Approaches to Text Categorization*. Journal of Information Retrieval (pp. 67-88), 1, 1/2.

Yang, Y. (2001). *A Study on Thresholding Strategies for Text Categorization*. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01).

Yavuz, T. & Guvenir, A. (1998). *Application of k-Nearest Neighbor on Feature Projections Classifier to Text Categorization*. In Proceedings of the 13th International Symposium on Computer and Information Sciences – ISCIS'98 (pp. 135-142), U. Gudukbay, T. Dayar, A. Gursoy, E. Gelenbe (Ed.), Antalya, Turkey.