ISSN:2249-5789

KNN Based Document Classifier Using K-d Tree: An Efficient Implementation

T.Priyanka[#1]
Dept. of Computer Science & Systems Engineering,
Andhra University, Visakhapatnam,
Andhra Pradesh, India
priyankacstau@gmail.com

N.Narasimha Swamy[*2]
Dept. of Computer Science & Systems Engineering,
Andhra University, Visakhapatnam,
Andhra Pradesh, India
narasimhaswamy999@gmail.com

Abstract

The existing techniques for text classification exhibits less efficiency on large collection of documents with higher dimensionality, especially when the corpus contains many noisy or irrelevant term features. To overcome these challenges, in this paper, the strengths of KNN and kd-tree are combined and we implemented the CenKNN classifier in two stages. The dimensionality of documents is reduced by projecting higher n-dimensional term feature space to lower l-dimensional class-centroid space using centroid classifier in the first stage. Later, the kd-tree search method finds the k nearest neighbours used in KNN classifier. The performance measured by F1-score on the Reuters-21578 dataset, has shown its effectiveness and reduced computation time.

1. Introduction

With the growing internet connectivity, the amount of digital information is becoming exponentially large day-by-day which may be either structured or unstructured. There is an enormous need to extract the useful or interesting knowledge from streaming text data available in electronic format such as in emails, news reports, web pages, scientific articles and micro-blogs. Therefore text categorization or classification ([11], [8]), has emerged as a contemporary research theme which is the process of assigning predefined classes to the text documents. Many challenges are being faced in text classification to deal with real-world applications such as news filtering and organization, spam detection, social media user classification etc., as it is essential to perform classification on a large-scale and high-dimensional text corpus containing imbalanced distributions [8], and irrelevant or noisy term features.

K-Nearest-Neighbor (KNN) [6], is a remarkable simple classifier and a popular instance-based learning method, which uses all the training documents for searching the nearest neighbors of test document to make a prediction. However, it is computationally expensive due to its lazy learning pattern and it suffers from class imbalance problem because of its majority voting classification method. Also, its success depends on availability of effective similarity measures and KNN is more sensitive to irrelevant or noisy term features. Choosing the appropriate parameter value of k for getting consistently favorable performance is a major challenge in KNN classifier.

Class-centroid-based linear text classifier [2], address the problems of imbalanced class distributions and also deals with bias towards majority class. In this classifier every class will be represented by its centroid, obtained by combining the prevailing term features of documents belonging to each class. So, class label of each centroid's class value is diverse and separable from others. The average weights of these term features in majority class are very smaller than those in minority class centroid. As a result, centroid classifier is less biased towards majority class. Also, Centroid classifier is able to scale up with data size. However, it misfits when the documents are not linearly separable by its boundaries within the class centroids.

K-d tree [12], is a binary search tree [4], for associative searching and stores the data in a k-dimensional space. To construct a k-d tree, the algorithm will simply choose one of the coordinate among the k dimensions to iteratively divide the remaining instances. To search for a given target instance, the search method will first finds a leaf node containing the target instance then, from that leaf node it will recursively roll back and searches its nearby nodes. When the distance to the next leaf node is not improved the searching is clogged.

In this paper, our contribution is implementing the CenKNN, a scalable and effective flat text classifier in R, for a large-scale high dimensional text corpus with mostly skewed class distributions and noisy or irrelevant term features.

We organized the rest of the paper as follows. Section 2 discusses about the related work and our implementing method is introduced in Section 3. Results are presented in section 4. The Paper is concluded with Section 5.

2. Related work

There are many methods to perform the text-classification task. Some of the existing text classifiers are discussed below.

A. KNN Classifier.

KNN ([3], [6], [7]), is lazy learning or an instance-based classifier, where the local approximation of the function is made and all the computation is deferred until classification. The straight forward method is, to assign majority class among the neighbors to the target instance or by distance weighted voting method [9], for assigning higher weight to the nearest neighbors via below formula (1),

$$y_t = arg\ max_{C_j} \sum_{d_i \in KNN^{d_t}} sim(d_t, d_i)\ I(d_i, C_j) \quad (1)$$

Where, KNN^{d_t} has the set of k nearest neighbors of test document, $sim(d_t, d_i)$ is the similarity between d_i & d_t, $ind(d_i, C_j)$ is the indicator function will be 1 or 0 depending on whether the d_i belongs to class C_j . KNN suffers from several drawbacks such as expensive computation time, sensitivity to imbalanced class distributions and parameter (K value) tuning.

B. Improvements on KNN.

For enhancing the efficiency of KNN, inverted index is used to pre-process the training documents. However, inverted index may not work well if the terms in the test documents are overlapped with many training documents.

By integrating data compression methods or dimensionality reduction methods with KNN, classification time can be saved. But this preserves only some of the term features and we may lose most of the critical information, resulting in accuracy degradation.

We can speed up the KNN classification by using fast search trees like k-d trees, B+ trees [14], SR-trees [15], and these methods also improves the classification accuracy. Still, these searching techniques are not efficient while classifying a large corpus in very high-dimensional space and it needs to spend more time for clustering the training corpus to constructing the search tree.

Moreover, for dealing with sensitivity to noise cluster-based methods have been proposed. But, clustering identifies neighbors on high-dimensional space which is more time-consuming.

Other methods have been explored for improving KNN sensitivity to text classification. Although, it's high computation time has not been reduced.

C. Centroid Classifier.

Centroid classifier [2], is a class-centroid-based linear classification technique, where each class is represented by its nearest centroid value. Centroid classifier is less likely biased towards the majority class. But, misfits for documents those are not linearly separable.

D. Rocchio.

Rocchio [13], works similar to the centroid classifier but uses prototype vectors for classifying the documents. Prototype vectors are calculated by the formula (2) below,

$$pv_j = \alpha \frac{1}{|C_j|} \sum_{d_i \in C_j} d_i - \beta \frac{1}{|D - C_j|} \sum_{d_m \in D - C_j} d_m \quad (2)$$

Where, pv_j is the class j's prototype vector. α and β adjusts the relative importance between positive and negative documents. If α=1 and β=0, Rocchio will be same as the centroid classifier. Rocchio cannot deal with the documents other than linear separable and unable to handle the classes with underlying sub-clusters rather than single cluster.

E. SVM.

The SVM [10], principle is to determine the support vectors for maximizing the margins of separation between the classes in a hyper-plane. SVM applies kernel methods such as polynomial and RBF kernels to achieve the desirable accuracy even on non-linearly separable data space. However, it is less effective on skewed data.

F. Naïve Bayes & Neural Networks.

These text classifiers are very simple, easy-to-implement and can achieve favorable classification performance in many applications. But has poor classification accuracy on skewed corpora and requires expensive computation in high-dimensional text documents.

3. Methodology

The methodology explains the clear depiction about the implementing classifier CenKNN [1]. The overview of the steps in our method are shown in the below schematic diagram.

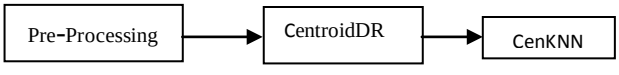

Fig. 1. Schematic Diagram of Classifier

Pre-processing is the prerequisite for classification of the text documents. After pre-processing, dimensionality reduction step can be performed via CentroidDR method which projects the *n*-dimensional documents space onto *l*-dimensional class-centroid-based space of *l*-classes. Finally, a k-d tree is constructed for searching the k-nearest neighbors among training documents over the test document. The Classifier detailed process is explained below:

A. Pre-Processing.

In the text classification tasks, Vector Space Model (VSM) [5], is commonly used for representing the documents. Given a set of N training documents D = {($\mathbf{d}_1$, y_1) , ($\mathbf{d}_2$, y_2) , · · · , ($\mathbf{d}_N$, y_N)} and a set of predefined classes y_i ∈ {C_1, C_2, · · · , C_l}, where each document is represented by a term-based vector as $\mathbf{d_i}$ = {t_1, t_2, · ·.,t_j · , t_n} ∈ R^n. n is the size of the vocabulary, consists of different terms in D and t_j is prominence of j[th] term estimated using TF*IDF.

B. Centroid Dimensionality Reduction Method (CentroidDR).

Motivated by the significant performance of Centroid, a class-centroid-based dimensionality reduction method *CentroidDR* has been introduced in *CenKNN* [1]. The assumption in CentroidDR is that, the documents are normally closer to their inherent class centroid rather than other centroids.

Centroid classifier is mainly successful due to the strong representation power of class centroids and good robustness to imbalanced class distributions and irrelevant or noisy term features. Another reason for its impressive performance is the cosine similarity measure which considers the term dependencies between class centroids and documents. CentroidDR uses class centroids and the cosine similarity measure to reduce the document dimensionality. It projects high-dimensional documents onto a low-dimensional space occupied by class centroids. In centroid, CentroidDR is with the combination of any linear classifier. Whereas, CenKNN replaces the linear classifier by a sophisticated non-linear classifier, for finding a better classification boundary in the class-centroid-based space.

After computing the centroids of all classes CentroidDR uses cosine similarity measure to map the documents into the class-centroid-based space. A class centroid can be generated efficiently by computing the mean vector of documents within the class, as given in below formula (3),

$$centroid_j = \frac{1}{|C_j|}\sum_{d_i \in C_j} d_i \qquad (3)$$

So $centroid_j$ represents the centroid of the class C_j, j = 1, 2, · · · , l. Projection of documents onto the new space can be performed using formula (4) below,

$$x_i^{(j)} = sim(d_i, Centroid_j) = \frac{<d_i, Centroid_j>}{||d_i|| \times ||Centroid_j||} \qquad (4)$$

Where j denotes the indices of the original documents d_i and the projected data After finding the similarities get the projected data D^* = {($\mathbf{x}_1$, y_1) , ($\mathbf{x}_2$, y_2) , · · · , ($\mathbf{x}_N$,y_N)} and each x_i is projected from d_i. This feature makes CenKNN as a scalable text classifier.

C. Text Classifier CenKNN.

Once we obtained the projected data on low dimensional class-centroid-based space, follow up with constructing the k-d tree for storing the projected data as it organizes the points in a k-dimensional space and search for its k nearest neighbors to classify the test document. Every non-leaf node generates a splitting hyper plane into half-spaces. Axis oriented splitting planes can be chosen by cycling through the axis as we move down (For example, in a 3-d tree the root is having the x-aligned plane, y-aligned planes for root's children, z-plane for root's grand children and again root's great-grandchildren would have x-aligned plane and root's great-great-grand children have all y-aligned planes)[16], or choosing the median of points that are being placed in the sub tree results in a balanced k-d tree having same distance from each leaf node to the root. K-d tree search method can be performed in CenKNN using Euclidean distance, a Minkowski metric, on the normalized space of the projected data. So, for finding the nearest neighbors on k-d tree CenKNN uses cosine similarity for effectiveness of the algorithm.

Algorithm (CenKNN)

Input: Consider D-set of training documents and d_t - test document

Output: Document d_t class label

(1) The centroid of each class is computed using formula (3)

(2) The similarities between the document and all of the centroids are computed for every document; Now assign these values to the dimension values of the projected data using formula (4)

(3) The projected data D^* = {($\mathbf{x}_1$, y_1) , ($\mathbf{x}_2$, y_2) , · · · , ($\mathbf{x}_N$,y_N)} is obtained. Every x_i is projected from d_i. The similarities obtained in the above step are the coordinates of the original documents in the projected space and their class labels are retained.

(4) Projected document vectors in D^* are to be normalized.

(5) Build a k-d tree on the obtained data from step 4.

(6) On the test document $d_t \in R^n$, project it on the class-centroid-based space and further normalize it. Thus we obtain $\hat{x}_t \in R^l$

(7) K nearest neighbors of $\hat{x}_t$ over the k-d tree are to be found.

(8) Using the KNN decision rule classify $\hat{x}_t$ according to the below formula(5).

$$y_{t=} arg\ max_{C_j} \sum_{d_i \in KNN^{\hat{x}_t}} (1 - dist(\hat{x}_t, \hat{x}_i))\, I(\hat{x}_i, C_j) \quad (5)$$

Where $KNN^{\hat{x}_t}$ has the set of $\hat{x}_t$ K nearest neighbors, $dist(\hat{x}_t, \hat{x}_i)$ is the Euclidean Distance between normalized test and training documents $(\hat{x}_t\ and\ \hat{x}_i)$ and $I(\hat{x}_i, C_j)$ denotes the indicator function which is either 1 or o depending on $\hat{x}_i$ belongs to class , C_j or not.

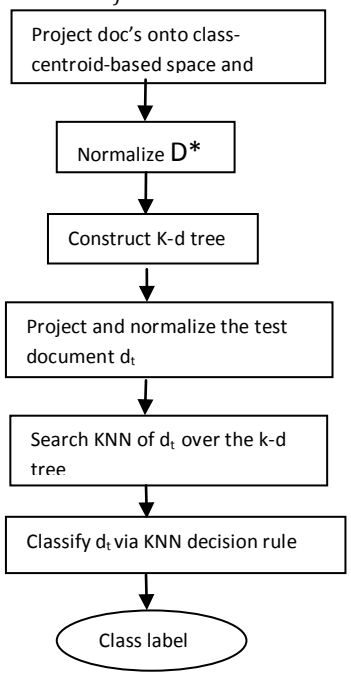

Fig. 1 Process Flow Diagram

4. Results

The proposed method CenKNN, by Guansong Pang et al. has been implemented and its performance has been evaluated by conducting experiments on Reuters-21578, a benchmark and standard corpus for text classification. We used one of the most popular split versions of Reuters-21578 called, "ModApte" Split.

Table 1. Classification Performance on Reuters-21578

	MICROF1	MACROF1
CenKNN	0.9841	0.9213
KNN	0.8950	0.8321
Centroid	0.9156	0.8431

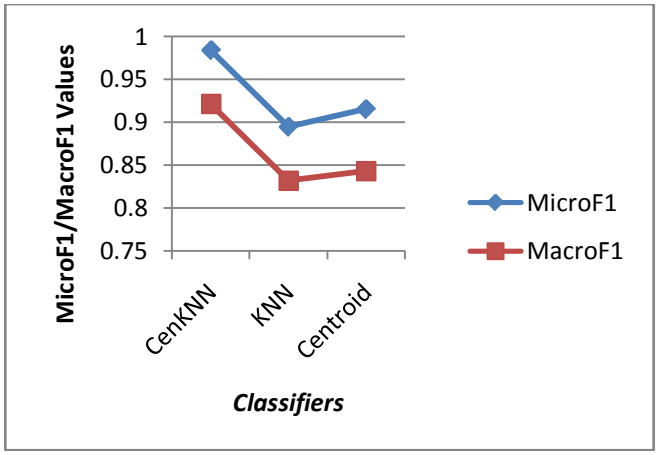

Fig. 2. Classification Performance

Precision and Recall can be used as evaluation measures for classification. By using these measures effectiveness can be calculated as F-measure or F-Score.

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

Now, F-Score can be computed using the below formula,

F1=(2*precision*recall) / (precision+recall)

As we are retrieving multiple class labels the general results can be viewed by averaging the evaluation measures: Micro-Average and Macro-Average. Global Calculation of F1 regardless of classes is the MicroF1. MacroF1 is simple the average over the F1 of all the categories.

Table 2. Accuracy of Classifiers on Reuters-21578

CENKNN	KNN	CENTROID
0.873	0.806	0.819

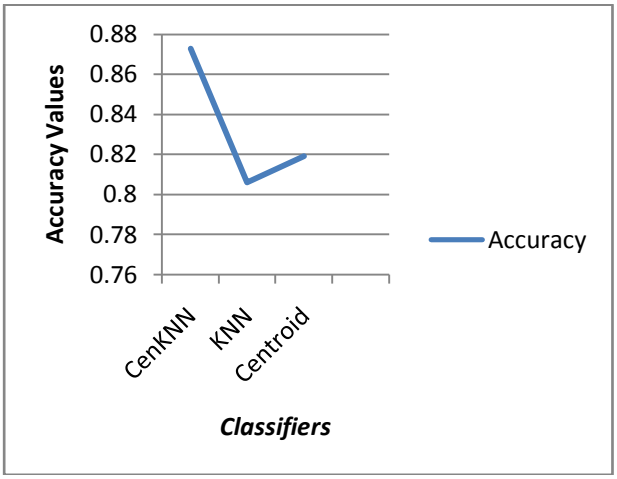

Fig. 3. Accuracy of Classifiers

Accuracy can be computed by summing the true positives over all the classes. Generally the

accuracy will be poor on skewed class distributions.

The performance of both CentroidDR and k-d tree search method is related with CenKNN for its efficiency and effectiveness. It has shown an improvement in classification accuracy over KNN and its variants by working on a lower-dimensional class-centroid-based space. Its superiority is clear while dealing with more complex corpora, such as R35 having more irrelevant or noisy term features where KNN cannot perform well. Actually, CentroidDR method is dependent on centroid classifier, since it uses class centroids to define similarity. But, CenKNN classifier predicts by using the KNN classification rule, makes the KNN very effective and to be less dependent on centroid. To ensure the efficiency in k-d tree search, a general rule that restricts the size of the training dataset N >> 2^k is imposed, where, k or l is the size of data dimensionality or number of classes. Still, slightly breaking the general search rule does not prevent CenKNN in obtaining preferable and desirable classification accuracy.

5. Conclusion & Future Scope

By integrating class-centroid-based dimension reduction method with the k-d tree search we have implemented the CenKNN classifier, proposed by [1]. As a result, by combining the strengths of both KNN and centroid classifiers it acquired good robustness to noisy or irrelevant term features and imbalanced class distributions. CenKNN is able to scale up with the data size because of its linear computational complexity. The classifier we implemented is very simple since it works on a lower-dimensional space and as a result we can notify greater classification accuracy compared to other well-known text classifiers. On the other hand, to obtain consistently favorable performance on any type of corpora, the parameter k is set to 10 by default. So, this eliminates the issue of choosing a well-suited k value in text classification. But, CenKNN is preferable only when dealing with lower value of K and small number of classes.

We can improve CenKNN to deal with underlying sub-clusters or more number of classes via hierarchical classification and class hierarchy. There is a scope to extend this work to the multi-label text classification tasks.

6. Acknowledgement

I would like to convey my gratitude to Prof.M.Shashi, my guide for her support all the way to the successful completion of the project research work.

7.References

[1] Guansong Pang. Huidong jin. Shengyi Jiang. CenKNN: A scalable and effective text classifier. Springer (2014)

[2] Miao Y., Qiu X. Hierarchical centroid-based classifier for large scale text classification. In: First Pascal Challenge on Large Scale Hierarchical Text classification (2009)

[3] Han X, Li S, Shen Z (2012) A k -NN method for large scale hierarchical text classification at LSHTC3. In :Third Pascal Challenge on Large Scale Hierarchical Text classification (2012)

[4] Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18 (9) : 509–517

[5] Salton G, Wong A, Yang C(1957) A Vector Space Model for automatic indexing. ACM 18(11):613-620

[6] Cunnigham P,Delany SJ(2007) k nearest neighbor classifiers UCD-CSI-2007-4

[7] Guo G, Wang H, Bell D, Bi Y, Greer K (2006) Using KNN model for automatic text categorization. Soft Comput 10(5):423-430

[8] Sun Y, Wong AK, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recog Artif Intell 23(4):687-719

[9] Tan S (2005) Neighbor-weighted k-nearest neighbor for unbalanced text corpus. Expert Syst Appl 28(4):667-671

[10] Joachims T (1998) Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning, pp.137 – 142

[11] Rajni J, Shwetha T (2012) Text Categorization – A Review. Elsevier(7) : 444-449

[12] Stefan Wass, Klaus-Dieter, Guido Derwand (2005) Using k-d trees to improve the retrieval step in case-based reasoning. Springer vol.837 : 167-181

[13] Joachims T (1996) A Probabilistic Analysis o f the Rocchio Algorithm with TFIDF f or Text Categorization.In : Proceedings of the 14th International Conference on Machine Learning , pp.143 – 151 (1996)

[14] Jagadish HV, Ooi BC, Tan K, Yu C , Zhang R (2005) iDistance: an adaptive B+-tree b ased indexing method for nearest neighbor search. ACM Trans Database Syst (TODS) 30(2):364–397

[15] Katayama N, Satoh S (1997) The SR-tree: an index structure for high-dimensional nearest neighbor queries. AC M SIGMOD R ec 26:369–380

[16] https://en.wikipedia.org/wiki/K-d_tree