

Improved k -NN Algorithm for Text Classification

Muhammed Miah

Department of Computer Science and Engineering
University of Texas at Arlington, TX, USA

Abstract - Over the last twenty years, text classification has become one of the key techniques for organizing electronic information such as text and web documents. The k -Nearest Neighbor (k -NN) algorithm is a very well known and popular algorithm for text classification. The k -NN algorithm determines the classification of new document by the class of its k -nearest neighbor. In this paper we propose an improved k -NN algorithm with a built-in technique to skip a document from training corpus without looking inside the document if it is not important, which improves the performance of the algorithm. It also has an improved decision rule to identify class from k -nearest neighbor to improve the accuracy by avoiding bias of dominating class with large number of documents. We conduct experiments on benchmark text classification datasets. The new and improved k -NN algorithm is suitable for other applications as well.

Keywords: data mining, k -nearest neighbor, text classification or categorization, text mining.

1 Introduction

During the last twenty years the number of text documents in digital form has grown enormously in size. As a consequence, it is of great practical importance to be able to automatically organize and classify documents. Research into text classification aim to partition unstructured sets of documents into groups that describe the contents of the documents. There are two main variants of text classification: text clustering and text categorization. The former is concerned with finding a latent group structure in the set of documents, while the latter (also known as text classification) can be seen as the task of identifying the appropriate class for a new document according to a group structure that is known in advance. In this paper we work on text categorization or classification where we try to improve the k -NN algorithm by improving the factors that affect the algorithm both in terms of performance and accuracy.

Text classification is the process of grouping texts into one or more predefined classes based on their content. The goal of text classification is the automatic assignment of documents to a fixed number of semantic classes. Each document can be in multiple, exactly one, or no class at all.

Document classification appears in many applications, including e-mail filtering, mail routing, spam filtering, news monitoring, selective dissemination of information to information consumers, automated indexing of scientific articles, automated population of hierarchical catalogues of

Web resources, identification of document genre, survey coding and so on. Automated text classification is attractive because manually organizing text document bases can be too expensive, or unfeasible given the time constraints of the application or the number of documents.

A number of statistical classification and machine learning techniques have been applied to text classification, including regression models, Bayesian classifiers, decision trees, nearest neighbor classifiers, neural networks, and support vector machines. As mentioned earlier, in this paper we focus on improving the k -NN algorithm which is a special case of nearest neighbor classifier. The k -NN algorithm has shown to be very effective in text classification as well as in other domains.

k -NN algorithm classifies a test document based on its k -nearest neighbor. The training examples can be considered as vectors in a multidimensional feature space. The space is partitioned into regions by locations and labels of the training samples. A point in the space is assigned to a class in which most of the training points belong to that class within the k nearest training samples. Usually Euclidean distance or Cosine similarity is used. During the classification phase, the test sample (whose class needs to be identified) is also represented as a vector in the feature space. Distances or similarities from the test vector to all training vectors are computed and k nearest training samples is selected. There are a number of ways to classify the test vector to a specific class. The classical k -NN algorithm determines the class with the majority voters from its k -nearest neighbors. Fig 1 provides an example [32] of how k -NN algorithm identifies a new sample to a class using multidimensional feature space from its k -nearest neighbor training samples. The test sample (circle in the middle) should be classified either to the first class of squares or to the second class of triangles. If $k = 3$ it is classified to the second class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$ it is classified to first class (3 squares vs. 2 triangles inside the outer circle).

k -NN is a case-based learning method, which examines all the training data for classification. The algorithm is not very well suited in some applications such as dynamic web mining for a large repository as it is a lazy learning method. This is because it has to examine the whole training data for classification as well as look at every keyword in a specific document. k -NN is a simple but effective method for classification. This motivates us to improve the algorithm in terms of efficiency and accuracy avoiding any bias from dominating classes.

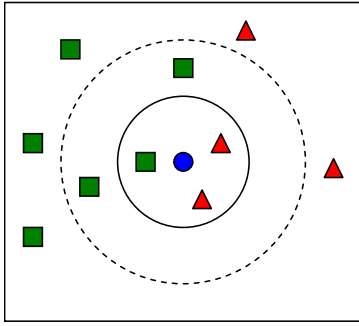


Fig. 1. Example of k -NN algorithm

There are three major factors that affect k -NN algorithm which are: 1) the similarity or distance measure used to find the k -nearest neighbor, 2) k , which is the number of nearest neighbor, and 3) the decision rule to identify a class for the test document from k -nearest neighbor. In this paper we try to improve the performance of the algorithm while improving the accuracy as well. We use the Cosine similarity measure which is a widely used and popular similarity measure in text classification. Like the classical k -NN algorithm, we do not consider all documents in the training corpus as well as try to avoid examining all keywords in a document to detect k -nearest neighbor fast. In fact, we propose a solution that may not need to go inside of each training document and also does not require examining all the keywords in a document, which makes the algorithm efficient. There is another problem in k -NN algorithm when a dataset is dominated by one or few classes that means when one or few classes contain most of the documents in the corpus and others have very few documents. In this situation, simply applying classical decision rule of k -NN algorithm which is selecting the class with most number of documents in k -nearest neighbor may mislead the result. To avoid this problem we use an improved decision rule to identify the right class from k -nearest neighbor for accurate classification.

In this paper we focus and conduct experiments for the improved k -NN algorithm in text classification domain, but this can be used in other applications as well such as e-mail filtering, mail routing, spam filtering, news monitoring, selective dissemination of information to information consumers, automated indexing of scientific articles, automated population of hierarchical catalogues of Web resources, identification of document genre, authorship attribution, survey coding and so on.

Main Contributions The main contribution in this paper may be summarized as follows:

1. We propose and develop an improved k -NN algorithm which is faster than classical k -NN algorithm while improving the accuracy.
2. We use a k -nearest neighbor buffer technique for which the algorithm can check very quickly whether it needs to go inside a training document to look at the keywords and skip the documents which are not important.
3. We impose improved decision rule to identify a class from k -nearest neighbor by giving more weight to the classes with higher number of training documents within

the k -nearest neighbor space and at the same time penalize the documents that are far from the test document to avoid the biasness by the class with large number of documents in training corpus.

4. We conduct experiments on three benchmark text classification datasets with different k values and evaluate the results.

The rest of the report is organized as follows. In section 2 we discuss some of the related works. Section 3 and 4 describe the algorithms and datasets used for the experiments respectively. In section 5 we discuss the data cleaning and preprocessing steps. Section 6 presents the results of the experiments. Section 7 is a short conclusion and then we provide the references in Section 8.

2 Related Work

Researchers have proposed a variety of algorithms for text classification purposes [1, 11, 13, 18, 21, 25, 26, 29, 30]. These are mainly on new algorithms and experiments were done how they perform compared to other algorithms. Some of them also compare existing algorithms [13, 28, 29].

As we mentioned before, there are many classification techniques have been proposed such as Naïve Bayesian (NB) classifiers [14], Decision Tree Classifiers [4], Decision Rules [16], Regression methods [29], Neural Network [22], k -NN classifiers [19, 29], Support Vector Machine (SVM) [10, 11], Rocchio classifiers [9, 20], etc. In applications with very large number of documents, efficiency is a vital issue.

Removing stop list (“a”, “the”, and so on) is also common in text classification as it contains meaningless words and can reduce the dimensionality of data. The practice of using a stop list is common and not normally considered for testing in the text mining process [1, 2, 13, 24, 26].

Stemming, a transformation method, converts words to their root form, is also a popular method for text classification. It reduces the number of distinct or unique words in a document by combining all forms of a word into one root form, which improve the efficiency. However, the reduction in unique words can affect the results of the overall process. Many studies report that the use of stemming will help categorization in some situations, but hurts it in other situations [5, 7, 12, 24]. Many researchers use stemming to improve efficiency.

Feature selection is also used in text classification [17, 21, 31] which is mainly use to reduce the dimensionality of data and improve scalability and efficiency.

[8] introduced very nice techniques to find approximate nearest neighbor which are also useful in text classification, but in this paper we focus on exact nearest neighbor, not approximate nearest neighbor.

The most related work to our paper would be text categorization using Weight Adjusted k -Nearest Neighbor (WAK -NN) text classification [6]. WAK -NN uses a weighted Term Frequency (TF) of keywords for similarity measure. But it did not consider avoiding looking inside the training documents which are not important. Also WA -KNN has an

additional computational cost for weight adjusted step of $O(cn^2)$ where c is the number of iterations in the weight adjustment step and n is the number of data points [6]. The decision rule used in *WA-KNN* is different than our paper. We give more preference to the classes with more number of documents to maintain the originality of k -nearest neighbor concept and at the same time penalize the documents which are far from the test document. *WAK-NN* just adds the similarity for each class on k -nearest neighbor and selects the one with the highest value.

3 Algorithms

As we mentioned earlier, in this work we intend to improve the k -Nearest Neighbor (k -NN) algorithm. We propose two improved versions of classical k -NN algorithms and compare with the classical k -NN algorithm. The brief fundamentals of the algorithms are given below:

3.1 Classical k -Nearest Neighbor Algorithm (k -NN)

The k -Nearest Neighbor algorithm (k -NN) is a method for classifying objects based on closest training examples in the feature space. k -NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification.

An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors. k is a positive integer, typically small. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes.

k -NN classifier is an instance-based learning algorithm that is based on a distance or similarity function for pairs of observations, such as the Euclidean distance or Cosine similarity measure. The cosine similarity is commonly used in Information Retrieval [23] and we used as the basic similarity measure in our algorithms.

The training corpus can be considered as an inverted list kind of narrow table (which is created during preprocessing step) with three columns {keyword, document, TF} where each row represents a keyword belongs to a specific document with corresponding Term Frequency (TF). Term Frequency (TF) is the frequency of the term or keyword in that document which is actually the number of times the keyword present in the document divided by the total number of keywords present in that document. We use TF in our algorithms and normalize within the document so that sum of all TFs in a document is equal to 1. Inverse Document Frequency (IDF) or combination of TF and IDF (TFIDF) also can be used. TFIDF is very popular in Information Retrieval domain. But in text classification, it was shown that TFIDF is not always very effective [3, 27].

The cosine similarity between two documents can be expressed as follows:

$$\text{Sim}(Q, D_i) = \frac{\sum_j w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}}$$

where Q is a query or test document, D is a training document relevant to Q and w are term frequencies of keywords. We refer $\text{Sim}(Q, D_i)$ as v in rest of the paper.

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques, for example, cross-validation. The special case where the class is predicted to be the class of the closest training sample (i.e. when $k = 1$) is called the nearest neighbor algorithm.

3.2 Improved k -Nearest Neighbor Algorithm (k -NN_I)

This is an improved version of k -NN algorithm. We use Cosine similarity measure to find k -nearest neighbor. We find the similarity between test and training documents using term frequencies and considering only the keywords present in both documents. To make the algorithm more efficient, we try to avoid examining the keywords in a document which are not important. That means we do not want to consider each and every keyword in a document, we only consider the important keywords for similarity measure. By important keywords here we mean the keywords which are present in both the test document and training document for which we need to measure the similarity. For example, if a test document Q has keywords $\{w_1, w_2, w_3\}$ and training document D_i has keywords $\{w_1, w_3, w_4\}$, then the cosine similarity value between Q and D_i would be calculated based on keywords w_1 and w_3 only. We also maintain a top- k buffer/array which contains the nearest k neighbors based on higher similarity value at any time to avoid considering the training documents which are not important for the classification of a specific test document. So the algorithm does not need to consider each document as well as each keyword inside the document which improves the performance significantly. The term frequencies of all keywords for each document of training corpus are calculated in preprocessing step. When a new test document comes as input, the algorithm calculates the term frequencies of all of its keywords. It also initializes a buffer/array of size k with empty value.

For the first k training documents the similarity measure goes into the top- k buffer/array which contains the document name or id and the similarity value found for each of the k document. After that for each training document it checks whether the sum of the term frequencies for all keywords in the document is greater than the minimum similarity value in top- k buffer or not. If it is greater then it looks inside the document, otherwise skip that document. The document can be skipped as the term frequency of a keyword is multiplied

with the term frequency of the same keyword in test document, and each term frequency has a value less than 1, so the sum of the multiplication never be greater than a value v , when simply the sum of the term frequencies is not greater than v .

```

D = {D1, D2, ..., Dn} is the set of training documents
Q is the new test document need to classify.
TF is the term frequency of a term in a document which is
preprocessed for training documents.
v is the cosine similarity value between Q and Di; (i = 1, 2, ..., n)

Algorithm: k-NN_I
Step1: Calculate the TFs of all keywords in Q
Step2: Initialize an empty buffer/array of size k, A[]
/* contains document id/name and similarity value with
the highest k cosine similarity, we call it top-k buffer */
Step3: int count = 0 /* Initialize a counter variable */
Step4: For each document Di in D
    If (count <= k)
        Calculate v between Q and Di
        /* considers only keywords present in both */
        Add document name and v into A[]
        count++;
    Else
        If (Sum(TFs) in Di > Min(v) in A[])
            Calculate cosine similarity value v between
            Q and Di
            /* considers only keywords present in both */
            If (new v > Min(v) in A[])
                Remove document with Min(v) from A[];
                Add document Di and new v into A[];
            /* Else Skip the document */
Step5: Return the class with highest # of documents in A[]

Algorithm: k-NN_I_DR
Step1: Perform steps 1-4 of algorithm k-NN_I
Step2: Calculate the Euclidean distance for each document in A[]
Step3: For each class present in A[], Calculate
    Cj = number of documents in A[] of class Cj *
    Sum(Euclidean distances of all documents in A[] of
    Class Cj)
Step4: Return the class with the highest Cj value

```

Fig. 2. Summary of k -NN_I and k -NN_I_DR algorithms.

For example, let consider a training document D_i , a test document Q , and a top- k ($k = 2$) buffer with similarity values of $A[1.5, 2.7]$. Here the current minimum similarity value in the buffer is 1.5. Assume D_i contains three keywords w_1 , w_2 , and w_3 with term frequencies of 0.2, 0.5, and 0.3 respectively. The summation of the term frequencies in D_i is 1, which is less than the minimum value 1.5 in the top- k buffer. In this situation the algorithm will skip D_i and will not look inside D_i for examining its keywords as the summation 1 is less than the minimum value 1.5 in the buffer.

This is true as we can see in the following example. Assume Q has two keywords w_1 and w_2 with term frequencies of 0.3 and 0.7 respectively. If we calculate the similarity measure of D_i and Q , the result would be $(0.2*0.3 + 0.5*0.7 =$

0.41) which is less than 1.5 and never can be in the top- k buffer. Remember, to calculate the similarity we only consider the common keywords both in D_i and Q . If the summation of all term frequencies in D_i is greater than the minimum value in the buffer, it may or may not go into the buffer. That's why we need to calculate the cosine similarity to check whether D_i goes into the top- k buffer or not. This is the step where the algorithm might avoid looking into the keywords of all documents that improve the performance. Fig 2 summarizes the steps in the algorithm to identify the class for a query or test document.

Maintaining the top- k buffer/array with highest k similarity values, and avoiding looking inside of each and every training document, the algorithm improves its performance. The algorithm follows the traditional k -NN decision rule to identify class for a test document by the majority votes from its k -nearest neighbor.

3.3 Improved k-Nearest Neighbor Algorithm with Improved Decision Rule (k -NN_I_DR)

Class with large number of documents in a corpus can bias the classification. Class with highest number of documents in k nearest neighbor may mislead the result if simply the test document is classified based on tradition k -NN decision rule (i.e., test document belongs to the class with highest number of documents in k -nearest neighbor). This is because it may happen that most of the time too many documents from the same large class will appear in the k -nearest neighbor list even they are far from the query point or test document while other documents from another class is very close but number is less.

So we optimize k -NN_I to avoid this biasness and improve the accuracy. We give more weight (preference) to the class with higher number of documents in top- k buffer (k -nearest neighbor) and penalize the documents which are far from the test document. Combination of the above two can avoid the biasness of large class. After we get the final top- k buffer, we calculate the actual Euclidean distance from the test document for each document in top- k buffer. Divide the summation of all Euclidean distances in top- k -buffer with the summation of all distances in a class within the buffer and multiply the new value with the number of documents in that class in the buffer. Then return the class with the highest value. This eliminates the biasness or domination of a large class. Fig 2 summarizes the steps in the algorithm to identify class for a query or test document.

4 Datasets

In this paper we use three widely used benchmark datasets for text classification, which are: 20_Newsgrpup, 4_Universities and Reuters-21578 datasets. We intend to work on different types of dataset, so we choose one dataset (20_Newsgrroup) which contains text documents, another dataset (4_Universities) which contains web pages, and the

last one (Reuters-21578) delimited by SGML tags. The descriptions of the datasets are given below:

4.1 20_Newsgrupup dataset

It is a well known data set for text classification. The dataset is a collection of 20,000 messages, collected from UseNet postings over a period of several months in 1993. Data are divided almost evenly among 20 different UseNet discussion groups or classes which are:

alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.windows.x, misc.forsale, comp.sys.mac.hardware, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc.

20_Newsgrroups dataset is taken from

<http://people.csail.mit.edu/jrennie/20Newsgrroups/> where data is sorted by date into training (60%) and test (40%) sets, does not include cross-posts (duplicates) and does not include newsgroup-identifying headers (Xref, Newsgrroups, Path, Followup-To, and Date).

4.2 4_Universities dataset

The dataset contains WWW-pages collected from computer science departments of various universities in January 1997. The dataset was collected by the World Wide Knowledge Base (Web->Kb) project of the CMU text learning group. The data were collected mainly from major 4 universities: Cornell, Texas, Washington, and Wisconsin, and that's why it is called 4_Universities dataset. The data collected from other universities are grouped as miscellaneous.

We take the data from the source

<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz>. The data are classified into seven different classes which are: Student, Faculty, Staff, Department, Course, Project, and Other.

The total number of web pages in corpus is 8278. Originally it was 8282, but 4 web pages were removed because of unrecognized symbols and characters in the pages. The number of web pages for each of the seven classes is: student(1640), faculty(1124), staff(136), department(182), course(930), project(504), other(3762).

The class other is a collection of pages that were not deemed the "main page" representing an instance of the previous six classes. (For example, a particular faculty member may be represented by home page, a publications list, a vitae and several research interests pages. Only the faculty member's home page was placed in the faculty class. The publications list, vitae and research interests pages were all placed in the other category.)

The data set contains pages from the universities are: Cornell (866), Texas (825), Washington (1205), and Wisconsin (1263). 4119 miscellaneous pages collected from other universities.

4.3 Reuters-21578 dataset

The dataset contains 21578 Reuters news documents from 1987. They were labeled manually by Reuters personnel. Labels belong to 5 different category classes, such as 'people', 'places' and 'topics'. The total number of categories is 672, but many of them occur only very rarely. Some documents belong to many different categories, others to only one, and some have no category. Over the past decade, there have been many efforts to clean the database up, and improve it for use in scientific research. We take the data from the source

<http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>. In order to divide the corpus into training and test sets, we use the modified Apte (ModApte) split [15]. With this split the training and test sets contain 9603 and 3299 documents respectively.

5 Data Cleaning and Preprocessing

After collecting the data from sources, we did some cleaning and preprocessing on all the datasets.

5.1 Data Cleaning

We clean the datasets and convert them into bags of words (text documents). After cleaning the datasets, they look like bag of words or terms or keywords. The brief overview of the cleaning process on the datasets is given below:

20_Newsgrupup dataset The datasets originally are text documents. We remove all the existing headers such as "From", "Subject", "Organization", and "Lines". The statistics of the dataset after cleaning are as follows: total number of documents is 18846, number of training documents is 11314 and number of test documents is 7532.

4_Universities dataset The dataset are originally web pages in html format, and after cleaning we converted it into text document. As like other dataset, here we also remove all the existing headers such as "MIME-Version", "Server", "Date", "Content-Type", "Content-Length", and "Last-Modified", and so on. The source of dataset suggested taking data from any one university to make as test data and not the part of data from all universities. So, we select data from Wisconsin as test data and data from other universities as training data. The statistics of the dataset after cleaning are as follows: total number of documents is 8278, number of training documents is 7015, and number of test documents is 1263.

Reuters-21578 dataset The data format is divided in 22 files of about 1000 documents delimited by SGML tags. We remove all tags, headers, etc. and convert them into text files of simple bag of words as like other datasets described above. The statistics of the dataset after cleaning: total number of documents is 12362, number of Training documents is 9063, and number of test documents is 3299.

For all the datasets, we remove the stop words but no stemming has been done. We also remove all the special

characters which are meaningless on text classification such as “>”, “<”, “,””, “.””, and so on.

5.2 Data Preprocessing

We do some preprocessing for the training corpus of all the datasets in advance and stored the data in database tables. We create inverted list like tables to calculate some variables in preprocessing step which are used in the algorithms such as: total number of documents in corpus, total number of documents per class, total number of distinct terms in corpus, total number of terms per class, term frequencies of each keyword for each class, term frequencies of each keyword for each document, and so on.

6 Experiments

In this section we measure (a) the accuracy (percent of test documents correctly classified), and (b) the performance (total running time to classify all the test documents in datasets) and evaluate the results.

System Configuration: We used a PC with Intel Xeon 2.00-GHZ, 3.25GB of RAM and 465GB HDD for our experiments. We implemented all algorithms in C#, and used Microsoft SQL Server 2005 RDBMS to store and access preprocessed data. We connected to the RDBMS through ADO. Text documents were used in normal .txt format.

Figs 3 and 4 respectively display the performance and accuracy of the algorithms on 4_Universities dataset. Similarly, Figs 5 and 6 display the performance and accuracy of the algorithms on 20_Newsgroup dataset, Figs 7 and 8

display the performance and accuracy of the algorithms on Reuters-21578 dataset.

As we can see in Fig 4 that accuracy does not have much effect on k -value for different algorithms on 4_Universities dataset. There is a very little improvement on the accuracy of the algorithm k -NN_I_DR when k value increases from 50 to 100, but this is not a significant change. Also there is no effect on accuracy of using improved decision rule particularly on this dataset. This is because the dataset is not bias by any single large class or few classes. But in Figs 6 and 8 we can see that the accuracy increases with increasing k value for all the algorithms. Also algorithm k -NN_I_DR provides more accuracy than other two algorithms. This is because there is some biasness of a single or few classes in the dataset even originally the documents were almost evenly distributed among all classes. Also increasing k -value increases the accuracy.

In terms of performance, we can see in Fig 3, 5 and 7 that algorithm k -NN_I outperforms algorithm k -NN for all the datasets as it can avoid looking inside some of the training documents. As we can see in Fig 5 and 7, for datasets with larger documents the algorithm k -NN_I improve the performance significantly. Algorithm k -NN_I_DR takes little more time to execute than algorithm k -NN_I because it does not just take the majority votes from its k -nearest neighbor to identify the class for the test document. It has to calculate the actual Euclidean distance for the k -nearest neighbor documents to impose the improved decision rule which can avoid any bias from large dominating class. With little more execution time the algorithm k -NN_I_DR can provide better accuracy in this kind of situation than algorithm k -NN_I.

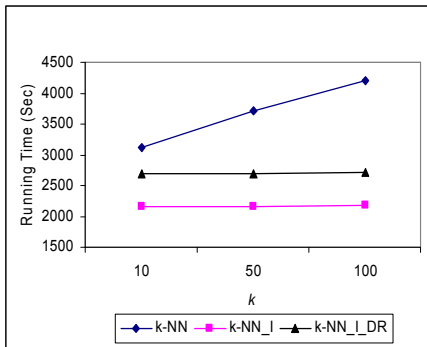


Fig. 3. Performance comparison on 4_Universities dataset with varying k value.

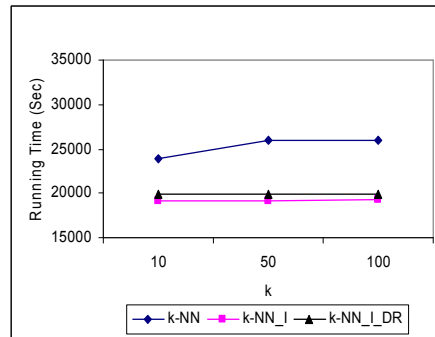


Fig. 5. Performance comparison on 20_Newsgroup dataset with varying k value.

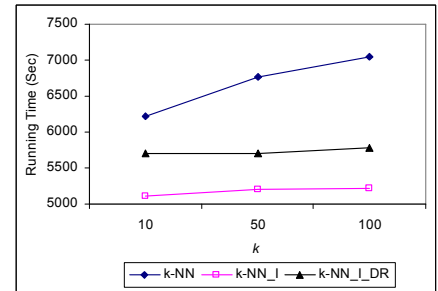


Fig. 7. Performance comparison on Reuters-21578 dataset with varying k value.

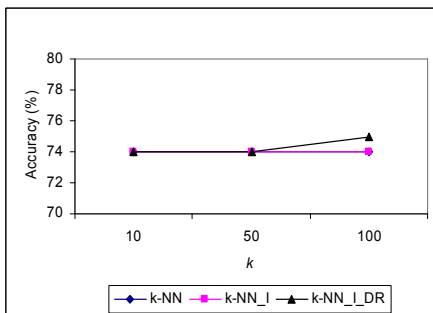


Fig. 4. Accuracy comparison on 4_Universities dataset with varying k value.

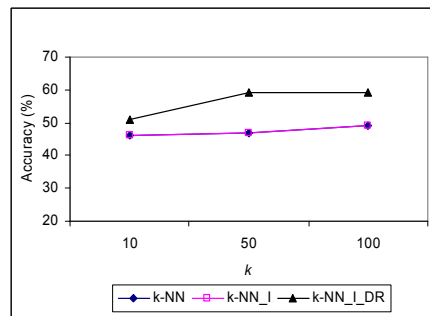


Fig. 6. Accuracy comparison on 20_Newsgroup dataset with varying k value.

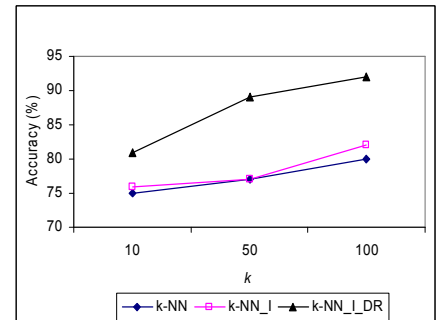


Fig. 8. Accuracy comparison on Reuters-21578 dataset with varying k value

Based on the discussion and experiments, it is clear that algorithm k -NN_I is faster than k -NN and k -NN_I_DR, and algorithm k -NN_I_DR is more accurate than the other two. So we can conclude that algorithm k -NN_I would be a better choice when the dataset is not bias by any single or few large classes that means there is no effect on the number of documents in a class that can affect the result of k -nearest neighbor. But when the dataset is bias by any large class, then definitely algorithm k -NN_I_DR would be the better choice. With little compromise on performance it can avoid such biasness and produce more accurate results. The accuracy measures might not seem very high from the experiments, this is because we do not use any indexing or stemming to improve the accuracy. We just want to compare the improved techniques with the classical k -NN technique. Use of indexing and stemming would improve the relative performance for the proposed techniques as well as for the classical k -NN technique.

7 Conclusion

In this paper our goal was to improve the classical k -NN algorithm in terms of performance and accuracy by considering couple of major factors that affect the algorithm. We imposed a top- k buffer technique that can skip looking inside each and every training document and also each and every keyword in a document, which improves the performance of the algorithm. We also proposed an improved decision rule to identify a class from k -nearest neighbor space by maintaining the classical k -NN property (majority votes) with penalizing the training samples which are far from the test sample, which can avoid any biasness from large dominating class in a dataset and improves the accuracy. This seems similar to weight adjusted k -NN approach but our technique is different as we discussed in related work section. We conduct experiments on three benchmark text classification datasets and evaluate the results.

In this work, our goal was not to compare our improved techniques with other state of the art text classification algorithms, but to improve the classical k -NN algorithm overcoming the factors that weaken the algorithm. In future we look forward to compare our proposed techniques with other text classification algorithm such as NB, SVM, Decision Tree, WA-KNN, etc. and improve the algorithms if needed such that they can outperform the existing algorithms both in terms of performance and accuracy.

8 References

- [1] Apte, C., Damerau, F., and Weiss, S., "Automated Learning of Decision Rules for Text Categorization." ACM Transactions of Information Systems 12.3 (1994): 233-251.
- [2] Blake, C., and Wanda Pratt. "Better Rules, Fewer Features: A Semantic Approach to Selecting Features from Text." ICDM 2001: 59-66.
- [3] Boley, D., Gini, M., Gross, R., Han, E. H., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and More, J., "Partitioning Based Clustering for Web Document Categorization", Decision Support Systems (1999).
- [4] Cohen, W. W and Singer, Y., "Context-Sensitive Learning Methods for Text Categorization", ACM Trans. Inform. Syst. (1999): 141-173.
- [5] Dave, K., Steve Lawrence, and David Pennock. "Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews." Proceedings of World Wide Web (2003): 519-528.
- [6] Han, H., Karypis, G., Kumar, V., "Text Categorization Using Weight Adjusted k -Nearest Neighbor Classification", PAKDD 2001: 53-65
- [7] Harman, D. "How Effective Is Suffixing?" Journal of the American Society for Information Science 42.1 (1991): 7-15.
- [8] Piotr Indyk, Rajeev Motwani: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. STOC 1998: 604-613
- [9] Joachims, T., "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization", Proceedings of ICML-97, 143-151.
- [10] Joachims, T., "A Statistical Learning Model for Text Classification for Support Vector Machines", 24th ACM International Conference on Research and Development in Information Retrieval (SIGIR) 2001.
- [11] Joachims, T., "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", Proceedings of 10th European Conference on Machine Learning, Chemnitz, Germany (1998): 137-142
- [12] Kao, A., and Steve Poteet. "Report on KDD Conference 2004 Panel Discussion Can Natural Language Processing Help Text Mining?" ACM SIGKDD Explorations Newsletter 6.2 (2004): 132-133.
- [13] Lewis, D., Ringuette, M., "A Comparison of Two Learning Algorithms for Text Categorization." In Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval (1994): 81-93
- [14] Lewis, D.D, "Naïve (Bayes) at forty: The independent assumption in information retrieval". Proceedings of ECML-98, 10th European Conference on Machine Learning (1998): 4-15.
- [15] Lewis, D. D.: Reuters-21578 Document Corpus V1.0. <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>
- [16] Li, H., Yamanishi, K., "Text Classification Using ESC-based Stochastic Decision Lists", CIKM-99, 8th ACM International Conference on Information and Knowledge Management (1999): 122-130.
- [17] Li, Z., Sun, M., "Scalable Term Selection for Text Categorization", Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 774-782, Prague, June 2007.
- [18] McCallum, A., and Kamal Nigam. "A Comparison of Event Models for Naïve Bayes Text Classification." Proceeding of AAAI/ICML-98 Workshop on Learning for Text Categorization (1998): 41-48.
- [19] Mitchell, T. M., "Machine Learning", McGraw Hill, NY (1996).
- [20] Rocchio, Jr. J. J., "Relevance Feedback in Information Retrieval. The SMART Retrieval System: Experiments in Automatic Document Processing", Editor: Gerard Salton, Prentice-Hall, Inc, Englewood Cliffs, New Jersey (1971).
- [21] Rogati, M., Yang, Y., "High-Performing Feature Selection for Text Classification", CIKM 2002, 659-661.
- [22] Ruiz, M. E. and Srinivashan, P., "Hierarchical Neural Networks for Text Categorization", SIGIR-99 (1999), 281-282.
- [23] G. Salton, "Automatic Text Processing", The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, 1989.
- [24] Waegel, D., and April Kontostathis. "TextMOLE: Text Mining Operations Library and Environment." Proceedings of the 37th SIGCSE technical symposium on Computer science education (2006): 553-557.
- [25] Weiss, S., Chidanand Apte, Fred Damerau, David Johnson, Frank Oles, Thilo Goetz, and Thomas Hampp. "Maximizing Text-Mining Performance." IEEE Intelligent Systems 14.4 (1999): 63-69.
- [26] Yang, Y. "An Evaluation of Statistical Approaches to MEDLINE Indexing." Proceedings of AMIA-96, Fall Symposium of the American Medical Informatics Association (1996): 358-362.
- [27] Yang, Y., and Chute, C. G., "An Example-Based mapping Method for Text Categorization and Retrieval", SIGIR-94 (1994).
- [28] Yang, Y., and Jan Pedersen. "A Comparative Study on Feature Selection in Text Categorization." ICML 1997: 412-420.
- [29] Yang, Y., Liu, X. "A re-examination of text categorization methods." Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1999): 42-49.
- [30] Zaiane, O., and Maria-Luiza Antonie. "Classifying Text Documents by Associating Terms with Text Categories." Proceedings of the thirteenth Australasian conference on Database technologies 5 (2002): 215-222.
- [31] Zheng, Z., Wu, X., Srihari, S., "Feature Selection for Text Categorization on Imbalanced Data", SIGKDD Explorations, 2004.
- [32] http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm.