Comparative Analysis of Document level Text Classification Algorithms using R

Maganti Syamala¹, Dr N J Nalini², Lakshamanaprameendra Maguluri³, Dr. R Ragupathy⁴
¹²³ Department of Computer Science and Engineering
¹ Dhanekula Institute of Engineering & Technology, Ganguru, Vijayawada, Andhra Pradesh 521139, India.
²³ K L University, Green Fields, Vaddeswaram, Guntur, Andhra Pradesh 522502 India.
² Faculty of Engineering and Technology, Annamalai University Annamalai nagar-608002, India.

Abstract: From the past few decades there has been tremendous volumes of data available in Internet either in structured or unstructured form. Also, there is an exponential growth of information on Internet, so there is an emergent need of text classifiers. Text mining is an interdisciplinary field which draws attention on information retrieval, data mining, machine learning, statistics and computational linguistics. And to handle this situation, a wide range of supervised learning algorithms has been introduced. Among all these K-Nearest Neighbor (KNN) is efficient and simplest classifier in text classification family. But KNN suffers from imbalanced class distribution and noisy term features. So, to cope up with this challenge we use document based centroid dimensionality reduction (CentroidDR) using R Programming. By combining these two text classification techniques, KNN and Centroid classifiers, we propose a scalable and effective flat classifier, called MCenKNN which works well substantially better than CenKNN.

Keywords: preprocessing in R, TF-IDF, Document based CentroidDR, CenKNN, MCenKNN, word cloud.

I. Introduction

Classification is a Machine learning task for predicting the value of class by building a model based on one or more categorical attributes or inputs. During past few decades’ numerous amount of information is available on internet and predicting the information is bottle-neck. To predict the information one of the most important technique progressively intensified in text classification, is to classify the documents into predefined classes.

Some of the Text classification algorithms are: Rocchio’s algorithm, K-Nearest-Neighboralgorithm (KNN), DecisionTree algorithm (DT), Naive Bayes algorithm (NB), Artificial Neural Network (ANN), SupportVector Machine (SVM) [5] and Voting algorithms. Analysis of Rocchio algorithm is very fast and easy to implement, which uses relevance feedback mechanism, but it has low accuracy due to linear combination. K-Nearest-Neighbor Algorithm’s basic principle is “tell me who are your neighbors, then I will tell you who you are”. At the beginning of 1970’s KNN [7] is a non-parametric lazy learning algorithm which was mainly used in statistical estimation and pattern recognition, more local characteristics of document are considered when compared against
Rocchio Algorithm. KNN is used in huge number of fields such as Nearest Neighbor based Content Retrieval, Gene Expression and 3D structure prediction.

The Reset of the paper is organized as follows: Section II. Preprocessing using R, Section III. CentroidDR, Section IV. MCenKNN, Section V. Experimental Results, Section VI. Conclusions, followed by References.

II. Pre-processing using R

Text mining plays a vital role in fast growing research area. Preprocessing techniques are applied on data set to reduce the size of the data set which will increase the effectiveness. Though it is somewhat time consuming process, but at the end it plays a very crucial role in terms of quality analysis. R is open source software developed at bell labs. Many statistical functions are built in and it is implemented in c, Fortran languages. It was inspired by S environment and extended via packages. Once you loaded all documents properly, we can process the documents by using tm package in R. tm package allows you to remove punctuation, capitalization, common words, numbers which decreases the text data and improve the efficiency of the system.

Converting to lowercase:
Suppose if we want a word to appear exactly the same every time it appears, then we need to change everything to lowercase by using the following process.

```r
docs<- tm_map(docs, tolower)
```

Removing “stop words”

“Stop words” are nothing but common words that usually have no analytic value. In every document, there are a plenty of those words (a, and, also, the, etc.), such words are frequent by their nature, and will stunt your analysis if they remain in the text.

```r
docs<- tm_map(docs, removeWords, stopwords("english"))
```

Removing Punctuations:

```r
docs<- tm_map(docs, removePunctuation)
```

Removing numbers:

```r
docs<- tm_map(docs, removeNumbers)
```

III. CentroidDR

Now-a-days high dimensionality reduction is a challenging task in handling massive amounts of data. Due to the rapid growth of the World Wide Web, there are several classification techniques, which aim to extract features by projecting the original real-world high-dimensionality data into a lower-dimensional space through algebraic transformations, but these techniques require a huge amount of memory and CPU resource. So we use linear classifier techniques called CentroidDR[6] document based Centroid Dimensionality Reduction. CentriodDR[2] is one of most efficient and scalable technique which basically projects high-dimensional documents into a low-dimensional space spanned by document centroids. CentriodDR[7] mainly uses document centroids to reduce the dimensionality of documents. Details of CentriodDR are given as follows:

**Algorithm 1 (CentriodDR)**

**Input:** Target set of training documents D.

**Output:** Projected data \( D^* = \{(X_1,y_1),(X_2,y_2),\ldots,(X_N,y_N)\}\), where \( X_i \in \mathbb{R}^l \), \( y_i \in \{C_1,C_2,\ldots,C_l\} \) for \( i=1,2,\ldots,N \).
Step 1: CentroidDR first computes the centroids of all documents. A document centroid is the mean representation vector of each document, as detailed in below formula, which can be generated very efficiently.

\[
\text{Centroid}_j = \frac{1}{|C_j|} \sum_{d_i \in C_j} d_i
\]

So centroid denotes the centroid of the documents \( C_i, i = 1, 2, \cdots, l \). and \( C_j \) represents the number of classes.

Step 2: Obtain the projected data \( D^* = \{(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)\} \). Each \( X_i \) is projected from \( d_i \) and \( Y_i \) represent their respective initial class-labels.

IV. MCEnKNN

KNN is an instance based learning, which is a part of supervised leaning that has been used in many fields like Machine Learning [7], Image processing, Pattern recognition. Disadvantage of KNN: it requires to pre-specify the value of the parameter \( K \). Computation cost is also high that need to compute distance between each query term instance to all training documents. KNN is not able to handle irrelevant attributes or noisy features, KNN [1] also suffers from class imbalance problems [3]. To handle this situation, this paper combines the best features of KNN and Document based centroid dimensionality reduction (CentroidDR) and named as MCEnKNN [4] which subsequently reduces the computation time and improve the efficiency compared with CenKNN. Here to classify the documents this proposed algorithm uses k-d tree which is similar essence of binary search tree to store the projected data and search the K nearest neighbors.

Algorithm MCEnKNN:

**Input:** Given a set of training documents \( D \) and a test document \( d_t \).

**Output:** The class label of document \( d_t \).

1. Project the high dimensional documents \( D \) onto a document-centroid-based space via CentroidDR, and obtain the projected data \( D^* = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\} \).
2. Normalize the projected document vectors in \( D^* \).
3. Build a k-d tree on the Normalized data.
4. For the test document \( d_t \in \mathbb{R}^n \), project it onto the document-centroid-based space, and normalize it, we then obtain \( \bar{x}_t \in \mathbb{R}^l \).
5. Search for the K nearest neighbors of \( \bar{x}_t \) over the k-d tree.
6. Classify \( \hat{y}_t \) based on the KNN decision rule, as detailed in formula below.

\[
y_t = a \quad m \quad c \quad \sum_{x_i \in K} x_i \left( 1 - d \quad (x_t, x_i) \right) l(x_t, c_j) = - - - - (2)
\]

Where \( K \) \( x_t \) denotes the set of K nearest neighbors of \( x_t \), \( d \quad (x_t, x_i) \) denotes the Euclidean distance.

V. Experiment Results

To analyze the performance of the document level text classification this paper proposed MCEnKNN which consider a real world text data set. A great way of applying text analysis towards your real data is to find TF-IDF of each word used. TF-IDF stands for Term Frequency - Inverse Document Frequency. It is a statistical measure used to evaluate how important a word is to a document.
Term-Frequency: is stated as number of times the term or word occurred in a document. Suppose consider three text documents (docs1, docs2, docs3) and one query document(query) querydocumentcontains”0. It has new new times. /@. <,” first document docs1 contains “1. It has New Yorktimes /@. <,” second document docs2 contains “2. /@. <, It has New York post” third document docs3 contains “3. It has loss Angeles/@. <, times”. For training data, assign the class-labels as (docs1, docs2)->New York and docs3->Loss Angeles and query->?

After loading all the target documents in R. it will discard the words like numbers, capitalization, common words, punctuation that do not contribute to distinguishing between the documents. Then it calculates the TF as follows:

\[
tf(t,d) = ft,d i.e. the number of times that term t occurs in document d.
\]

For all the documents, we calculate the tf scores for all the terms in document against query document as follows:

![Fig 1:Term frequency matrix](image)

From the above Fig 1 it is clear that zero in every document represent that the particular term has never appeared in that document and non-zero represent number of times that particular term has appeared in that document.

Inverse document frequency: Estimate the scarcity of a term in the whole document collection. (If a term occurs in all the documents of the collection, its IDF is zero.).

\[
t(t, D) = \log \frac{N}{\{d \in D: t \in d\}}
\]

Where N is number of documents,\{d \in D: t \in d\} the number of documents in which particular term appears and it is shown as follows:

![Fig 2: Weighted TF-IDF matrix](image)

From the above Fig2 we multiplied the tf scores by the idf values of each term, obtaining the above matrix of documents-by-terms:
We now compute the centroids of all documents which we define it as CentroidDR(Centroid Dimensionality Reduction) i.e.; Project the high dimensional documents D onto a document-centroid-based space. A document centroid is the mean representation vector of each document, as detailed in below formula.

$$\text{Centroid}_i = \frac{1}{|C_i|} \sum_{d_i \in C_i} d_i$$ \hspace{1cm} (1)

So centroid\textsubscript{i} denotes the centroid of the documents C\textsubscript{i}, i = 1, 2, \cdots, l. and C\textsubscript{j} represents the number of classes.

From the above Fig 3 the obtained centroids of each and every document including query document which is known as projected data D* = \{(x\textsubscript{1}, y\textsubscript{1}), (x\textsubscript{2}, y\textsubscript{2}), (x\textsubscript{3}, y\textsubscript{3}) \cdots, (x\textsubscript{N},y\textsubscript{N})\} where x\textsubscript{1},x\textsubscript{2}, x\textsubscript{3},...x\textsubscript{N} are the centroid values and y\textsubscript{1}, y\textsubscript{2}, y\textsubscript{3},...y\textsubscript{N} are the initial class-labels of their respective documents.

Finally, we classify the class-label of test document \hat{x}_t by using KNN decision rule and Search for the K nearest neighbors of \hat{x}_t over the k-d tree. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote.

$$y_t = \arg \max_{m} \sum_{x_i \in \mathcal{K}} m \left( 1 - \text{dist}(\hat{x}_t, x_i) \right) I(x_i, C_j)$$ \hspace{1cm} (2)

Where, KNNx\textsuperscript{*}tdenotes the set of K nearest neighbors of x\textsuperscript{*}t,\text{dist}(x\textsuperscript{*}t, x\textsuperscript{*}i) denotes the Euclidean distance between x\textsuperscript{*}iandx\textsuperscript{*}t, and I(x\textsuperscript{*}i, C_j) is the indicator function, which is 1 when x\textsuperscript{*}ibelongs to C j otherwise 0.

Next, we use Fast k-nearest neighbor distance searching algorithms like kd-tree by setting the default value of parameter K=10. Through various experiments conducted the proposed method MCenKNN got stable performance than KNN where K is set to be a default value with 10.
Fig 5: class-label of test document prediction using kd-tree

From the above Fig 5, it is clear that the query belongs to the class “New York” i.e.; it is more similar to the docs1 which was also predicted using KNN decision rule. When MCenKNN is compared with CenKNN, MCenKNN which uses centroid values spanned by their documents rather than using class-centroids. This MCenKNN will predict the class-label of test document accurately and takes less time when compared to CenKNN.

Generally humans are strong at visual analytics. That is part of the reason why “Plot Word Frequencies” and “word-clouds” have become so popular.

| TECHNOLOGY          | Comparative Analysis |
|---------------------|----------------------|
|                     | KNN Classifier       | Centroid DR | KNN+Centroid DR | MKNN |
| R Statistical Computing | ✓                     | ✓            | ✓                | ✓    |
| MATLAB              | ✓                     | ✓            | ✓                | ×    |
| Scikit-Learn        | ✓                     | ✓            | ✓                | ×    |
| Erdas image         | ✓                     | ✓            | ✓                | ×    |
| Envil               | ✓                     | ✓            | ✓                | ×    |
| Weka                | ✓                     | ✓            | ✓                | ×    |

Table 1. Serial tool and their analysis

Accuracy: Accuracy refers to the closeness of a measured value to a standard or known value. It helps us to think (visually) using confusion matrix as shown below in table 1.

|                  | CORRECT | NOT CORRECT |
|------------------|---------|-------------|
| SELECTED         | TP      | FP          |
| NOT SELECTED     | FN      | TN          |

Table 2: confusion matrix

In this confusion matrix, the “correct” cells are:

- **TN**: the number of true negatives
- **TP**: the number of true positives

“Error” cells are:

- **FN**: the number of false negatives
- **FP**: the number of false positives
This example shows that out of 3 documents docs1, docs2 belongs to “New York” class and docs3 belongs to “Loss Angeles” class. And now to search the terms in a sentence that belong to “Loss Angeles” but more number of documents that are retrieved belong to “New York” class. Then the probability of the terms in a sentence that it is required to search(relevant) is only 10% i.e. 10 but the irrelevant terms are about 99% i.e. 99,990.

Therefore, Accuracy is defined as:

Accuracy=99,990/1,00,000=99.99%

We make us of Precision and Recall which are two measures to define test’s accuracy. Precision: % of selected items that are correct. Recall: % of selected items that are correct.

Precision and Recall are defined using the formula’s:

Precision = TP / (TP+FP) ------(3)
Recall = TP / (TP+FN) ------(4)

Therefore, the computed precision and recall values of the taken example is:

Precision=1/(1+1)=50% and recall=1/(1+1)=50%

As, a small example it became equal but if tested by taking large dataset it gives more recall compared to precision and also it is noticeable that as soon as the recall value get increased the precision value get decreased.

A measure which combines both precision and recall is the harmonic mean of precision and recall i.e.; nothing but the traditional F-measure or balanced F-score and is also defined as the precision and recall when they are close, and is more generally the square of the geometric mean divided by the arithmetic mean.

VI. Conclusion

This paper proposes A Modified Document text classifier MCenKNN using R to eliminate imbalanced class distribution problems and noisy-term features. Through our experimental results it shows that by combining strengths of two text classification algorithms like Centroid and KNN. This paper proposes Centroid based text classification algorithm which can handle high dimensional data spanned by document centroids and skewed data i.e., non-separable data by taking the advantage of non-linear text classifier like K-Nearest Neighbor. MCenKNN which uses centroids spanned by document centroids is able to classify the class-label of test document accurately when compared to centroids spanned by classes in CenKNN. Therefore the experimental results which are obtained using MCenKNN are more accurate when compared to CenKNN.

References:

[1]. Tan S (2006) An effective refinement strategy for KNN text classifier. Expert SystAppl 30(2):290–298.
[2]. Tan S, Cheng X (2007) An effective approach to enhance centroid classifier for text categorization. In:Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases, pp. 581–588 (2007).
[3]. Sun Y, Wong AK, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern RecogArtifIntell 23(4):687–719.
[4]. Wang X, Zhao H, Lu B (2011) Enhance k-nearest neighbour algorithm for large-scale multi-labeled hierarchical classification. In: Second Pascal Challenge on Large Scale Hierarchical Text classification (2011)

[5]. Wan CH, Lee LH, Rajkumar R, Isa D (2012) A hybrid text classification approach with low dependency on parameter by integrating K-nearest neighbor and support vector machine. Expert Syst Appl 39(15):11880–11888

[6]. Wang X, Zhao H, Lu B (2013) A Meta-Top-down Method for Large-scale Hierarchical Classification. IEEE Trans Knowl Data Eng, 99 (2013). doi:10.1109/TKDE.2013.30.

[7]. Guansong Pang · Huidong Jin · Shengyi Jiang (2014) CenKNN: a scalable and effective text classifier. Data Min Knowl Disc DOI 10.1007/s10618-014-0358-x.

[8]. Lakshmana Phaneendra Maguluri A comparative analysis of clustering based segmentation algorithms in microarray images in International Journal of Emerging Science and Engineering 2013