Large Scale Hierarchical Text Documents Classification using Machine Learning Algorithms: KNN, SVM, RF AND NB

Y. Sailaja[1], Dr. G. Lavanya Devi [2] and K. Srinivasa Rao[3]

[1]*M.Tech, Department of Computer Science and System Engineering, Andhra University College of Engineering, Visakhapatnam, India*

[2]*Assistant Professor, Department of Computer Science and System Engineering, Andhra University College of Engineering, Visakhapatnam, India*

[3]*Research Scholar, Department of Computer Science and System Engineering, Andhra University College of Engineering, Visakhapatnam, India*

[1]*(E-mail: yadlapallisailaja@gmail.com)*

[2]*(E-mail: lavanyadevig@yahoo.co.in)*

[3]*(E-mail: sri.kurapati@gmail.com)*

Abstract:

Hierarchies are becoming more popular for the organization of the documents, particularly on the Web. Wikipedia and RCV1 are the examples where there are millions of documents that are classified into multiple classes in hierarchy fashion. Along with their widespread use, it rises an interesting problem of automating the classification of new documents to the classes of the hierarchy. Large Scale Hierarchical Text Classification is one of the classification problems, it has thousands of classes and millions of documents and those are connected in parent and child relation. As the size of the dataset increases, the number of documents to be classified also increases. So, the increasing of dataset leads to the data sparsity problem. Therefore, this poses a challenge to classify the data in hierarchy manner. To overcome this problem, different machine learning methods (KNN, Random Forest, SVM, Navie Bayes) are used based on the text classification. We implement and compare these methods for the better performance for classifying hierarchical data.

Keywords: *Text Classification, Hierarchical Data, Multi label Classification, kNN, Random Forest, Navie Bayes and SVM algorithms.*

1. Introduction:

Text classification or categorization is a theme based classification; it classifies the natural language texts. Text classification deals with the problem of assigning documents to a predefined set of classes. It has been used in several real world applications. Those are the online news classification, spam filtering and organization of large scale web pages. In machine learning a classifier perform this procedure automatically using the positive and native instances. In case of simple prediction of classifier the results are rarely correct.

In this paper, studied a more complex case in which he large scale test can run the number of classes into thousands. The number of classes can run into thousands in large scale text classification. In some cases each document belongs to single class in other cases it may belongs to more than one class. Another aspect of the problem is that the classes are connected through the parent child relation composing a hierarchy. Text classification includes several algorithms; those are Random Forest, Support Vector Machine (SVM), Navie Bayes, k-Nearest Neighbour (kNN), etc[2-5].

Among those Random Forest algorithm is one of the simplest classification algorithms [4]. It is very most used algorithm when compares to other algorithms. The purpose of this Random Forest algorithm is to separate the documents into classes and predict the class of new sample document.

Many user-centric and content-driven web applications have been growing over the past few years. Some examples of these applications are blogs, wilds and resource sharing systems. The need to organize such unsystematic or unorganized content into categorized resources has driven the motivation for text categorization.

The main contributions of this research paper are:

- Predict the labels of given a document and its term frequencies.
- Experimental analysis has been carried out on dataset of Reuters Corpus Volume I (RCV1) and the proposed models achieved superior results over standard approaches.

The overall structure of this research paper is as follows. Section 2 presents related work. Proposed approach is given in Section 3. Section 4 discusses about the implementation of the proposed models. Section 5 reveals the results. Finally, Section 6 gives conclusions along with future scope of the work.

2. Literature Survey:

Sowmya et al. have studied multilabel classification problem that can be used in several real world applications such as protein function classification, music classification and semantic classification of images. In multilabel classification each document belongs to the more than one category. Most of the researchers are attracted towards multilabel classification on text categorization.

Multilabel classification can be classified into two different groups: Problem transformation, and Algorithm adaptation.

These methods are helps in modifying the multilabel classification task into many single-label classifications, regression or label ranking tasks. Bag-of-words, Binary Relevance and TF-IDF are the specific learning algorithms to manage multilabel data directly by next group of methods. The most commonly- used problem transformation is TF-IDF scheme. It gives the weight of word w in the document d.

TF-IDF Weight (w, d) = Term Frequency (w, d) X log (N/ DocFreq(w)).

Rohit Babbar et al. addressed new bounds on the generalization errors of classifiers deployed in large-scale taxonomies. These bounds make explicit the trade-off that both flat and hierarchical classifiers encounter in large-scale taxonomies and provide an explanation to several actual findings reports. This introduced such bounds is first time and that gives explanation of the behaviour of flat and hierarchical classifiers is based on theoretical grounds. They also propose a well-founded way to select nodes that should be pruned so as to derive a taxonomy better suited to the classification problem. They use here a simple pruning strategy that modifies the taxonomy in an explicit way.

3. Model Design:

Firstly, take the dataset from the sklearn (scikit- learns) .The dataset is in the csv form. Then organise the dataset into train and test dataset then store it. Now calculate the TF-IDF values for every respective document in the dataset i.e., train and test datasets. Apply different machine learning algorithms then calculate the accuracy for each classification algorithm and prove Random Forest algorithm is having the best accuracy when comparative with others.

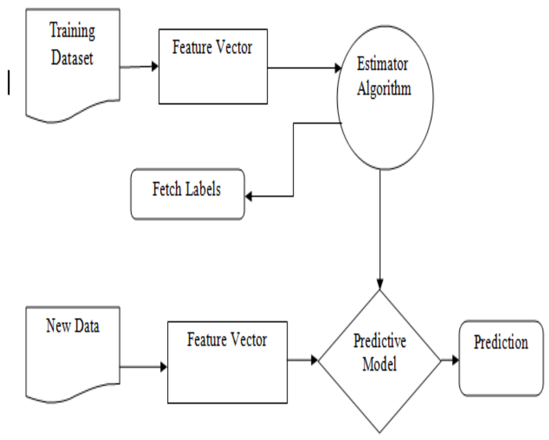

Figure 1: Model Design

Proposed model consists of two phases. Those are training phase and testing phase. In training phase we have to collect a fixed number of samples documents for each concept and then merge it. The above process results in a super document which have to be pre-processed and then indexed by using TF* IDF method. This method represents each concept by the centroid of the training set. Now index the new documents by using a vector-space method and then make comparison between the document vector and centroid for each concept then finally classify the documents.

To implement the algorithm we are going through the phase step by step workflow.

Training phase:

In training phase we follow four steps and those are mentioned below:

- Loading dataset
- Splitting data into train and test datasets
- Calculating feature vector
- Training classifier with Python scikit learn

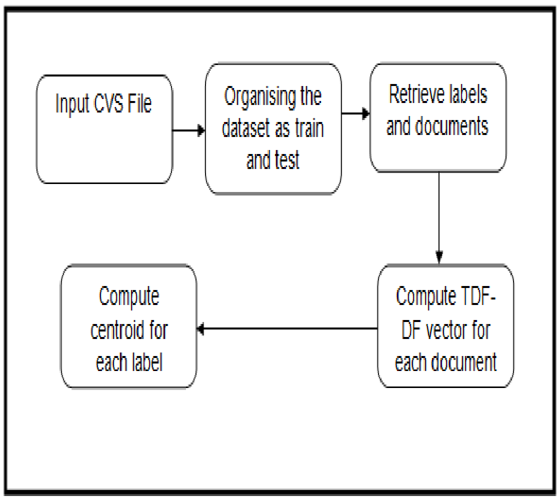
Input CVS File
Organising the dataset as train and test
Retrieve labels and documents
Compute TDF-DF vector for each document
Compute centroid for each label

Figure 2: Training Architecture

Loading dataset:

The dataset used was RCV1 (Reuters Corpus Volume I).The dataset is imported from the scikit-learn. It is in the form of CSV where each line corresponds to an instance. Here is an example:

6 0:41 8:1 18:2 54:1 442:2 3784:1 5640:1 43501:1

The first number (6 in the example) represents the category of the instance. In which each pair of numbers separated by '**:**' .In these first number represents the features id and the second number is its frequency.

Characteristics of dataset:

Classes	103
Samples total	804414
Dimensionality	47236
Features	real, between 0 and 1

Splitting dataset:

In scikit-learn a random split into training and test sets can be quickly computed with the **train_test_split** function. The training dataset is using supervised learning methods to train the model. The test dataset provides an unbiased evaluation of a final model to fit on the training dataset. The data in the train dataset should never use in test dataset.

Calculate TF-IDF:

TF-IDF standards for "Term Frequency - Inverse Data Frequency". In this the term frequency gives the frequency of the word in each document in the data set.

The inverse data frequency gives the weight of rare words across the entire documents. Using these two we can find out TF-IDF.

Training Classifier:

Machine learning has number of models. This paper mainly focused on some algorithms those are Naive Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbour (KNN), Random forest . We can import these algorithm from the scikit learn on train dataset to predict the test dataset.

Testing architecture:

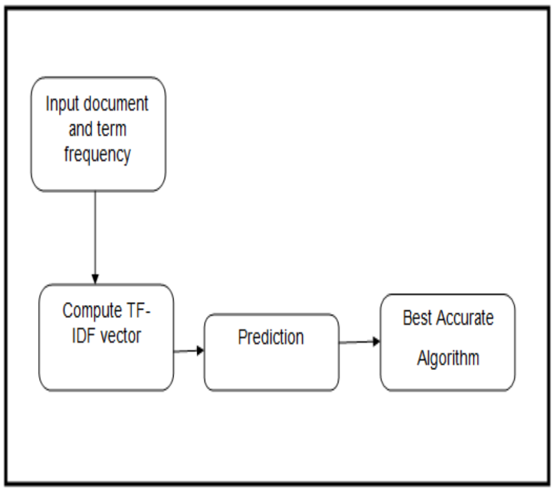

Figure 3: Testing Architecture

Perform Prediction:

In this we perform prediction for predicting the dataset which are related to the new document for the future usage. This predicted dataset is used to build a model. That predictive model is then used on current dataset to predict what will happen next or to suggest actions to take for optimal outcomes.

Accuracy calculations:

Using classifications algorithms to fit the predicted data, this would run our data over the current data and gives us the accuracy of the classification. Accuracy is the ratio of number of correct predictions to the total number of input samples.

Accuracy = Number of correct predictions / Total number of predictions made

4. Implementation:

In first phase we can import the dataset from scikit-learn and store it as training dataset then we compute the TF-IDF values for each of the document in the dataset as follows:

- First we calculate the term frequency it gives the frequency of each word in the each document in the dataset. TF formulae are given below:
  TF(t) = (Number of times term t appears in a document) / (Total number of terms in a document)
- Secondly we calculate the inverse data frequency it gives the weight of rare words across the entire documents. IDF formulae are given below:
  IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

The computation of Feature vector is independent of each other, there is no comparison on the accuracy of prediction in classifying the documents. Using Jupiter notebook with python3, the model is implemented.

Navie Bayes algorithm:

1. Load the dataset
2. Calculate the mean and standard deviation for each attribute, by class value of the training data.
3. Make predictions using the above obtained results.
4. Finally we get the accuracy of the model we have created.

Support Vector Machine algorithm:

1. Load the dataset.
2. Organise the dataset using the train_test_split() function.
3. Fit the model on train dataset using fit() function and perform prediction on the test dataset using predict() function.
4. Get the accuracy by comparing actual test dataset values and predicted dataset values.

KNN algorithm:

1. Load the dataset
2. Organising the dataset as train and test.
3. Training the predictions ort
4. Evaluating the algorithm and get the accuracy.

Random Forest Algorithm:

1. Load the dataset.
2. Train the model with the dataset.
3. Tune the model and create a model with optimized values.
4. Fit the mode and calculate the accuracy.

5. Results:

The Calculated accuracies of four algorithms i.e., KNN, SVM, Random Forest and Navie Bayes are 0.69, 0.81, 0.85, 0.77. Figure 4 shows that Random Forest is the most efficient algorithm to classify the hierarchical text documents among four algorithms.

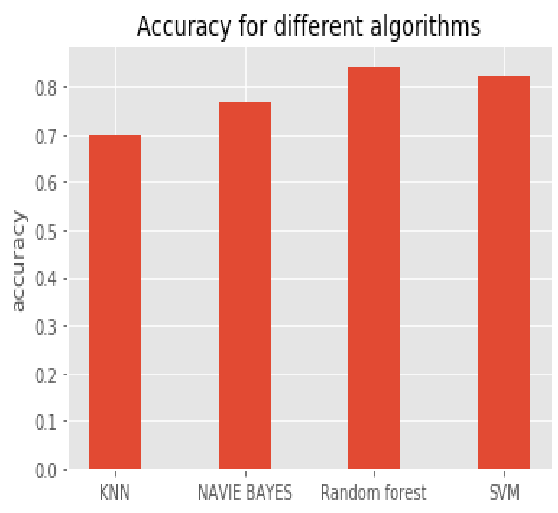

Figure 4. Accuracy for different Algorithms

6. Conclusion:

We proposed a vector space architecture for the prediction of the labels of a given documents. We compared four algorithms namely k-Nearest Neighbours, Support vector machine, Navie Bayes and Random forest algorithm. We elaborated why Random forest algorithm is the best, accurate and also most efficient among the three other algorithms. Regardless of the primitive nature of these algorithms, the drawbacks are simply negligible when considering the massive scale of the dataset. This work can be extended by deep learning techniques.

References

[1]. Kaggle LSHTC Challenge, htlp://kaggle.comlc/lshtc

[2]. Sowmya B J 1 Chetan1 K.G.Srinivasa2 “Large Scale Multi-label Text Classification of a Hierarchical Dataset using Rocchio algorithm“ 2016

International Conference on Computational Systems and Information Systems for Sustainable Solutions 978-1-5090-1022-6/16/$31.00 ©2016 IEEE.

[3]. Jong-Yeol Yoo1 and Dongmin Yang1 IY Yoo,D Yang" Classification Scheme of Unstructured Text Document using TF-IDF and Naive Bayes Classifier" Advanced Science and Technology Letters Vol.111 (COMCOMS 2015),pp.263-266.

[4]. Myungsook Klassen and Nikhila Paturi "Web document classification by keywords using random forests "California, USA 91360.

[5]. Chin HengWan, Lam HongLee, Rajprasad Rajkumar, DinoIsa "A hybrid text classification approach with low dependency on parameter by integrating K-nearest neighbor and support vector machine", 10.10.16.eswa.2012.02.068 .2012

[6]. Xiao Li, Da Kuang, Charles X. Ling

"Active learning for hierarchical text classification"Volumepart-1, kualampur malasiya, may 29,2012.

[7]. Rohit Babbar, Ioannis Partalas, Eric Gaussier, Massih-Reza Amini "On Flat versus Hierarchical Classification in Large-Scale Taxonomies" BP 53 - F-38041 Grenoble Cedex 9.

[8]. Basu, Tivar" Effective Text Classification by a Supervised Feature Selection Approach "2012 IEEE 12th Internatinal conference on data mining workshops pages 918-925 december 10-10,2012.

[9]. CH Wan, LH Lee, R Rajkumar, D Isa - Expert Systems with Applications.

[10]. M. Sujatha, G. Lavanya Devi, K. Srinivasa Rao and N. Ramesh, "Rough Set Theory Based Missing Value Imputation", Cognitive Science and Health Bioinformatics. Springer Briefs in Applied Sciences and Technology. Springer, Singapore, Online ISBN 978-981-10-6653-5, 2018.

[11]. Diman Ghazi, Diana Inkpen, Stan Szpakowicz, "Hierarchical versus Flat Classification of Emotions in Text" Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text", pages 140–146,Los Angeles, California, June 2010.

[12]. Xiaogang Han, Shaohua Li, and Zhiqi Shen "A k-NN Method for Large Scale Hierarchical Text Classi_cation at LSHTC3" Conference Paper · September 2012