



FIT 5149 Applied Data Analysis

**Data Analysis Challenge –
Classification of documents**

Shashank Sharma 28906446

Kaushal Vuppala 29100801

Nitya Mehra 29030552

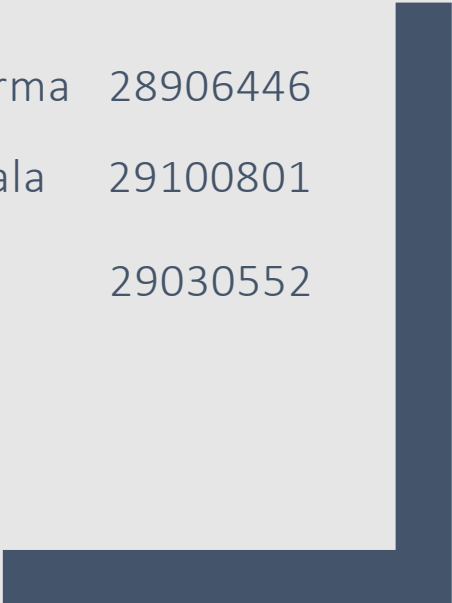


Table of Contents

1. Introduction.....	2
2. List of R-Libraries used	2
3. Pre-Processing of documents	2
4. Feature Selection and Generation	3
5. Building Classifier and prediction	3
6. Description of SVM Classifier.....	4
7. References.....	4

1. Introduction

In this data analysis challenge, we perform multi-class classification on a large set of news articles using **SVM Classifier**. The news articles were split into two sets randomly, out of which 80% are training documents and 20% are testing documents. The class labels are provided for the training documents and the objective is to predict the class labels for the testing documents. The total number of training class labels provided is 23, which makes the task a multi-classification problem.

Now, we need to build a supervised learning model which predicts the labels for the testing documents. Before the model is built, we need to pre-process the training and testing documents data for the generation of input features of the model. The feature generation is done using tokenization, case normalization, removal of stop words, removal of the most/less frequent words and error reduction techniques. This pre-processed data is used to generate and select an optimum set of features using unigrams, bigrams, and TF-IDFs. We try to find the best features which give us the best possible classification testing accuracy. Now, these features are passed as input to the SVM classifier and the model is trained using the training data. Once the model is built, we predict the class labels for the testing data.

2. List of R-Libraries used

We have used various libraries in this task. The description of the libraries is mentioned below.

- **tm**: This package is used for text mining.
- **RWeka**: This is an R interface to Weka, which is a collection of machine learning algorithms and provides tools for pre-processing, classification, regression, clustering, association rules, and visualization.
- **dplyr**: This package is used for manipulating data frame objects.
- **e1071**: This package contains functions for various classifiers like SVM.
- **randomForest**: This package is used to perform classification and regression based on a forest of trees using random inputs.

3. Pre-Processing of documents

In order to classify the news articles based on the text present in it, we had to ensure that only the words with reasonable importance to the news articles are fed to the model. Failing to do would produce a model with either really high variance (in case of an overfit) or really high bias (in case of an underfit). Therefore, pre-processing is the most important section of building a classifier. The pre-processing task can be divided into 3 parts:

Tokenization: Tokenization is the process of splitting long strings of text into smaller chunks of data such as phrases or individual words called tokens. Tokens can be alpha-numeric, all alphabetical or all numeric, therefore, not all tokens are not useful for building a model which depends on the nature of the analysis. We have removed all the tokens that were only numerical as it was redundant information as it would have offered no assistance to classify the articles.

Normalization: It is very important to maintain consistency within data for building a classifier. In any language, a word with the same root meaning can be written in many forms depending on tense, gender, plurality etc. Therefore, while working on text data, it is important to reduce words to its root or base form to maintain consistency. In this project, we used a simple process called stemming which cuts the end or beginning of a word based on a list of common prefixes and suffixes that can be found in an inflected word. For example: if a word ends with 'ed' or 'ing' then cut it from the word. We have also transformed every word to lower case to maintain consistency in the data.

Noise Reduction: At this stage, we want to ensure that every word that cannot contribute to the classifier is removed from the corpus. 'Stopwords' are words which are less important for any type of text processing, therefore all the stopwords were removed from the data. Also, all the punctuations and white spaces were removed from the words in the corpus. As high sparsity was observed in the document term matrix, all the words which did not appear in more than 95% and less than 5% of the documents were removed.

4. Feature Selection and Generation

It is important to select an optimum subset of features to be used for model construction in order to ensure that the model works in reasonable time as the time taken to build a model is directly proportional to the number of features used. Also, it helps to reduce the chances of overfitting the data as an optimum subset of features is selected. To select an optimum subset of features, we constructed a document-term matrix of unigrams and bigrams which contains the count of occurrence of each n-gram in each document present in the corpus.

We used term frequency-inverse document frequency (TF-IDF) to represent the occurrence of words in document term matrix as TF-IDF is one of the most useful methods to reflect how important a word is to a document in a corpus. It increases proportionally to the number of times a word appears in a document and inversely to the number of documents in the corpus that contain the word. Therefore, it takes into consideration that some words are used more than others but are not of any significance. We also normalized the TF-IDF of all the features present in each document to remove skewness in each document. This was done by scaling it such that the mean is zero and the standard deviation is one for every feature.

Hence, we generate combined features for both train and test data i.e. the total number of features in train and test data are similar. In total there are four feature files (unigrams and bigrams separately) generated for both test and train data. These files are used as input feature files to build the classifier.

5. Building Classifier and prediction

After the generation of the features, we pass these features as input in the model. We selected the best features which could give us the best possible classification accuracy. We have tried many supervised learning classifiers like SVM, Random Forest, Naïve Bayes Classifier, LDA, QDA etc. for this problem. Of all the classifiers, SVM gives the best possible classification accuracy.

The library `e1071` is used for building SVM classifier. The input file for this classifier is the combined unigrams and bigrams feature file for train and test data. The class labels for the test data are not provided, so we split the training data 4:1 ratio into the sample train and test data. The model is built on sample train data using 'svm' function from the library mentioned previously. After the model is built, using 'predict' function the class labels are predicted for the sample test data. We generate confusion matrix for the sample test data and calculate the testing accuracy, precision, recall, and F1-score. The results for the built classifier are:

Testing Accuracy: 74.5%

Testing Precision: 0.75

Testing Recall: 0.75

Testing F1-score: 0.74

Now the model is built again using the whole data without any splits and the class labels for the test data are predicted. This might improve the accuracy of the model since we have used the whole data to build the model.

6. Description of SVM Classifier

SVM's are known as Support Vector Machines, which are supervised learning models with learning algorithms that analyses and predict data used in regression and classification analysis. However, it is widely used for classification analysis. SVM is trained with a series of data which is classified into two categories. The model is initially trained using this classified data. The objective of the SVM is to determine to which category a new data point belongs to. Hence, SVM is a non-binary linear classifier. The SVM algorithm has margins between the classified categories and it maps the sorted data with these margins as far as possible. The major applications of SVM are text and hypertext classification, image classification, and hand-writing recognition.

We plot each data point in an n-dimensional space where n is the total number of features. Each feature has a value of a specific coordinate. We find hyper-planes (margins) that differentiates multiple classes as far as possible. The coordinates of each data point are known as support vectors. So, the SVM algorithm segregates the data into multiple classes.

7. References

1. CRAN - Contributed Packages. (2018). Retrieved from <https://cran.r-project.org/web/packages/>

2. Support vector machine. (2018). Retrieved from https://en.wikipedia.org/wiki/Support_vector_machine
3. What is a Support Vector Machine (SVM)? - Definition from Techopedia. (2018). Retrieved from <https://www.techopedia.com/definition/30364/support-vector-machine-svm>
4. Learning, M., & code), U. (2018). Understanding Support Vector Machine algorithm from examples (along with code). Retrieved from <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
5. Risueño, T. (2018). What is the difference between stemming and lemmatization?. Retrieved from <https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/>
6. Tf-idf. (2018). Retrieved from <https://en.wikipedia.org/wiki/Tf-idf>
7. Mayo, M. (2018). A General Approach to Preprocessing Text Data. Retrieved from <https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html>