Text Classification using Support Vector Machine(SVM)

Karankumar Bhaskarbhai Patel
Student Id: 1111107
Lakehead University

Bhavik Subhashchandra Ray
Student Id: 1111419
Lakehead University

Dhairya Shaileshbhai Patel
Student Id: 1110149
Lakehead University

Pankti Rakeshbhai Patel
Student Id: 1111102
Lakehead University

***Abstract*—The given literature describes about text classification which is being implemented on the dataset namely BBC(British Broadcasting Corporation) articles full text and category. This dataset is bifurcated into 5 sub-categories which are tech, business, sport, entertainment and politics. This dataset is available on kaggle. For implementation of text classification we have used SVM. Some pre-processing steps are carried out to remove irrelevant data. By performing appropriate hyper-parameter tuning we obtained some optimum parameters. These optimum parameters were responisble to get maximum accuracy. We received 98% accuracy in our trained model.**

***Index Terms*—Text classification, GridSearchCV, Natural Language Processing(NLP), SVM**

I. INTRODUCTION

With the introduction of the World Wide Web(WWW), the volume of data on the internet has grown tremendously. Although, such a vast collection of information is valuable because much of this information is a text and it becomes a problem for humans to recognise the most important information. The classification of text helps to resolve this challenge. Text mining is one field that derives value from unstructured data, and it is concerned with producing quality information from unstructured text. It processes in such a way that computers and statistical models can consume, with the objective of identifying patterns and knowledge to derive value. Text classification also known as text tagging or text categorization is the process of categorizing text into organized groups. By using NLP, text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content[1]. The traditional methods for classifying text include mainly k-Nearest Neighbor (k-NN), Decision Trees, Linear Regression (LR), Naive Bayes (NB), SVM etc. Their selection of features is based mainly on the bag-of words (BoW), n-grams, and Term Frequency-Inverse Document Frequency(TF-IDF). In our project we have used the SVM for the implementation of Text Classification.

II. BACKGROUND

A. Process of Text classification

Text classification is one of the important and typical task in supervised machine learning. Assigning categories to documents, which can be a web page, library book, media articles and gallery[6]. Fig 1 demonstrates the architecture of Text classification. In text classification initially the dataset is loaded. We have splitted the dataset into training and testing sets with the ratio of 70:30. Subsequently, pre-processing steps like removal of stop words, removal of punctuation, Stemming, Labelizing and Tokenization are carrried out. After the features are extracted we get the classification results.

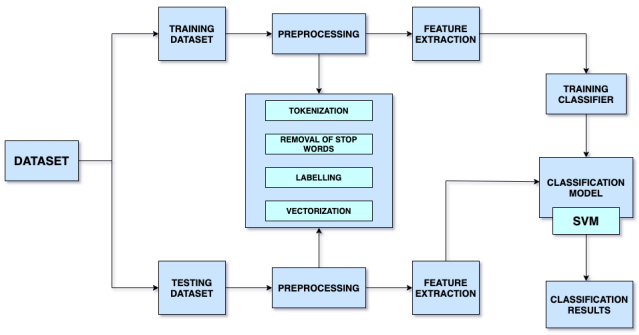

Fig. 1. Architecture of Text Classification

B. Data Pre-processing

Data Pre-Processing is the method of converting raw data into an efficient and useful format[2]. Also it can also be stated as the process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. The Pre Processing steps which are included in the Text Classification are as follows:-

- Removing Stop Words
- Removing Punctuation
- Labelizing
- Stemming
- Tokenization
- Text Vectorization

1) Removing Stop Words: In natural language processing, useless words (data), are referred to as stop words[4]. I have removed stop-words(which include “the”, “a”, “an”, “in”) because these words don’t waste space in my database, and don’t take up valuable processing time. I can remove them easily, by storing a list of words that you consider to be stop words. I have used Natural Language Toolkit(NLTK) in python because it has a list of stop-words stored in 16 different languages.

2) Removing Punctuation: Punctuation won’t get removed while removing the stop words. Other than stop words, there various characters which should be removed. Punctuation removal is important as it may affect to the output also because the meaning of the sentence differs by using some punctuation. They can be removed manually. In my model I have removed these punctuation- ? : ! ; ( ) “ - . ,

3) Stemming: Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers[3]. To perform stemming NLTK library is used so that derivatives of the words are removed and inflexion is also eliminated. ’PorterStemmer’ is being imported from nltk.stem to perform stemming on text.

4) Labelizing: As the dataset is bifurcated into 5 different categories. While performing text classification labels are assigned to each category of the text which is given in the dataset. To assigne the lables we have used the LabelEncoder. It is used to normalize the labels and transforms non-numerical lables(as long as they are hashable and comparable) to numerical labels. LabelEncoder can be used to normalize labels. Also, it converts labels back to original encoding.

5) Tokenization: Tokenization is the process of breaking down a sequence of strings into words, keywords, phrases and symbols called tokens. In this process the punctuation are discarded. Thereafter, these tokens become input for another process like parsing and text mining. To perform tokenization we have imported nltk.tokenize. When the text is tokenized they are converted into numbered index.

6) Text Vectorization: Machine learning algorithms take the numeric feature vectors as input. Thus, when working with the text documents, we need a way to convert each document into a numeric vector. This process is known as text vectorization[5]. We have used the TF-IDF technique to perform the Text Vectorization. TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. The outcome is a combination of two metrics: how frequently a term occurs in a document, and the inverse variable frequency of the term across a list of documents.

For record search and material retrieval, TF-IDF has been invented. It operates by growing proportionally with the amount of times a term occurs in a document, however the number of documents including the term is compensated. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don’t mean much to that document in particular.

Expression to find TF-IDF: To find the TF-IDF weights, there is a formula which is explained as below.

$$W_{i,j} = tf_{i,j} \times log(\frac{N}{df_i}) \quad (1)$$

Where,

$W_{i,j}$ = TF-IDF weight for token i in document j
$tf_{i,j}$ = Number of occurrences of token i in document j
df_i = Number of documents that has token i
N = Total number of documents in the training corpus

III. LITERATURE REVIEW

From the comprehensive literature survey conducted in the field of text classification, it has been noted that the commonly used data mining methods are SVM, k-NN, NB, Artificial Neural Networks(ANN), Rocchio algorithm and Association Rule Mining (ARM). These methods, along with the number of papers using these methods, are shown in Table I. As shown in the Table I, SVM is the most commonly used researcher in their work. Most of the authors have worked on the SVM algorithm and have suggested its improved version to improve the applicability of this algorithm, thereby enhancing the performance of text classification. k-NN algorithm is the second common method used by researchers. It has been noted that 86% of papers used machine-based learning methods and just 6% of papers used statistical methods. Of the numerous machine learning algorithms studied in literature, it has been noted that SVM and k-NN algorithms are the most commonly used machine learning algorithms in the field of text classification. 65% of the articles used such algorithms. Although some of the authors have also used statistical methods, their use has been very limited in the last few years as these methods are black box approaches and are highly data dependent. It is exciting to see a dramatic change from traditional statistical methods to modern machine learning methods[11].

TABLE I
MOST IMPORTANT DATA MINING METHODS USED

RANK	DATA MINING METHODS	PAPERS
1	SVM	55
2	k-NN	31
3	NB	23
4	ANN	10
5	Rocchio Algorithm	09
6	ARM	04

IV. PROPOSED MODEL

A. Dataset

We have used the BBC articles full text and category dataset which consists of 2225 entries. In our model we have splitted the dataset into training and testing sets according to the ratio of 70:30. This ratio of training and testing gave our model best accuracy. Fig 2 displays the graph of news categories with respect to the count. The dataset is further classified in 5 categories of news. They are as follows:-

- Tech
- Business
- Sports
- Entertainment
- Politics

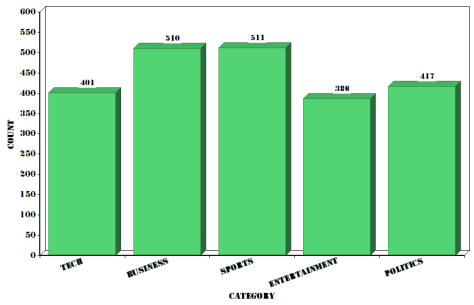

Fig. 2. Category vs Count

B. Libraries

To perform text classification necessary libraries need to be imported. We used the pandas library to load the data and perform the pre-processing steps. Numpy is used to perform the mathematical calculations. We have also used libraries such as nltk and sklearn. Sklearn library supports the SVM algorithm whereas the nltk library is used for performing text processing as it contains libraries for tokenization, stemming, tagging, parsing and classification.Below mentioned is the list of libraries used:

- numpy
- sklearn
- pandas
- nltk
- matplotlib

C. Pre-processing the data

Initially, labelizing was performed. Each category of news was allocated a unique label. After that stop words and punctuations were removed in order to execute efficient classification. Tokenization is carried out to break sequence of strings and tokens are obtained. At the end, Text vectorization is performed which converts text into numeric vector.

D. Proposed SVM model

A SVM is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they are able to categorize new text. SVM is a fast and dependable classification algorithm that performs very well with a limited amount of data[7]. SVM is an algorithm which determines the best decision boundary between vectors that belong to a given group or category and vectors which do not belong to it. The code for SVM with SVC is given in appendix section X.A. This can be applied to any kind of vectors which encode any kind of data which means in order to leverage the power of SVM text classification, the texts have to be transformed into the vectors[8]. The objective of a Linear Support Vector Classifier(SVC) is to fit to the data which is being provided, returning a best fit hyperplane that divides, or categorizes the data.

E. Use of GridSearchCV

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. We used Grid Search to find the best kernel. We have used two attributes. The actual code for GridSearchCV is given in appendix section X.C and D.

- best_estimator_: estimator
- n_splits_: int

The estimator which was chosen by search gave us highest accuracy and minimum loss whereas the n_splits is number of cross-validation splits(fold/iterations).

F. Tuning Hyperparameters

a) Kernel: The primary function of the kernel is to transform the given dataset input data into the required form. There are various types of functions such as Linear, Polynomial, and radial basis function (RBF) in which Polynomial and RBF are useful for non-linear hyperplane as they compute the separation line in the higher dimension. The code for obtaining best kernel is given in appendix section X.B. In some applications, it is suggested to use a more complex kernel which can separate the classes that are curved or nonlinear. As a result, this transformation can lead to more accurate classifiers[9].

b) Regularization: Regularization parameter in python's Scikit-learn C parameter is used to maintain regularization where C is the penalty parameter, which represents an error term. The task of error term is to tell the SVM optimization that how much error is bearable. Through this the trade-off between decision boundary and misclassification term is controlled. The smaller value of C creates a small-margin hyperplane whereas the larger value of C creates a larger-margin hyperplane[9].

c) Gammma: Over-fitting issue is caused when a lower value of Gamma will loosely fit the training dataset and the higher value of gamma will exactly fit the training dataset. It means that a low value of gamma considers only nearby points in calculating the separation line whereas the higher value of gamma considers all the data points in the calculation of the separation line[9].

G. Confusion Matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm. The accuracy of classification alone can be misleading if there exists an unequal number of observations in each class or if there are more than two classes in the dataset. Calculating a confusion matrix can give a better idea that whether the classification model is correct and what types of error are generated.

V. EXPERIMENTAL ANALYSIS AND COMPARISON

While implementing the text classification we carried out various experimental analysis by altering some parameters in order to get highest accuracy. We compared the existing model with the proposed model. It is observed from Table II that SVM model gives 98% accuracy which is better than the CNN model. CNN uses Tokenization and Labelizing as pre-processing whereas SVM uses Stemming, Labelizing, Tokenization, Vectorizing using TF-IDF. Simple looping is used to search the hyper parameters in existing model whereas GridSearchCV is used in SVM.

TABLE II
COMPARISON BETWEEN EXISTING AND PROPOSED MODEL

FEATURES	EXISTING MODEL	PROPOSED MODEL
Model used	CNN	SVM
Pre-Processing steps	Tokenization and Labelizing	Stemming, Labelizing, Tokenization, Vectorizing using TF-IDF
Searching Hyperparameters	Used simple looping	used GridSearchCV method
Accuracy	0.955	0.98

VI. RESULTS

By using 2-gram TF-IDF along with 5 cross-validation, we got final results in which the weighted average was 0.98 for Precision, Recall and f1-score and support was 668. The final values of Precision, Recall, f1-score and support are shown in Table III.

TABLE III
FINAL RESULTS

Precision	Recall	f1-score	Support
0.97	0.98	0.98	148
0.99	0.97	0.98	111
0.97	0.97	0.97	128
0.99	1.00	1.00	171
0.97	0.97	0.97	110

VII. APPLICATIONS

Text classification brings consistency and flexibility to the table. It is amazing to see how Advertisers, Product Managers, Designers, Academicians and Engineers can all make use of this technology. The entire idea of technology is to make life simpler. Classifying broad text data helps standardise the interface, makes searching simpler and more appropriate, and enhances user experience by simplifying navigation. Platforms such as E-commerce, news agencies, content curators, blogs, directories, and likes can use automated technologies to classify and tag content and products. Text classification can also be used to automate CRM task and to automate and speed up this process. The experimental results show that SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly. With their ability to generalize well in high dimensional feature spaces, SVMs eliminate the need for feature selection, making the application of text categorization considerably easier. Another advantage of SVMs over the conventional methods is their robustness[10]. SVMs show good performance in all experiments, avoiding catastrophic failure, as observed with the conventional methods on some tasks.

VIII. CONCLUSION

Text classification is performed on the dataset called BBC articles full text and category. The trained model received the accuracy of 98% after estimating the optimum parameters. When the dataset is loaded initially it is splitted into training and testing sets according to ratio of 70:30. Data-preprocessing is also done in which steps like Stemming, Labelizing, Tokenization, and Vectorizing using TF-IDF are used. Also, we used the GridSearchCV in order to get appropriate kernel. We used some attributes in hyper parameter tuning which resulted in good accuracy for our model. We have used 2-gram TF-IDF along with 5 cross-validation which resulted into maximum accuracy. Thus, we concluded that using SVM proved to be efficient for implementation of Text Classification

IX. GROUP CONTRIBUTION

Table IV displays the contribution of each group member in this project.

TABLE IV
CONTRIBUTION

NAME	TASK	CONTRIBUTION
Dhairya	Coding and Report	25%
Karan	Coding and Report	25%
Bhavik	Coding and Report	25%
Pankti	Coding and Report	25%

REFERENCES

[1] https://monkeylearn.com/what-is-text-classification/
[2] https://www.geeksforgeeks.org/data-preprocessing-in-data-mining
[3] https://www.geeksforgeeks.org/python-stemming-words-with-nltk/?ref=lbp
[4] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[5] https://www.kaggle.com/edchen/text-vectorization
[6] https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a
[7] https://monkeylearn.com/blog/introduction-to-support-vectormachines-svm/
[8] https://monkeylearn.com/text-classification-support-vector-machines-svm/
[9] https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
[10] https://dzone.com/articles/text-classification-applications-and-use-cases
[11] Jindal, Rajni, Ruchika Malhotra, and Abha Jain. "Techniques for text classification: Literature review and current trends." webology 12, no. 2 (2015).
[12] https://www.kaggle.com/yufengdev/bbc-text-categorizationHyperparameter-tuning

X. APPENDIX

A. SVM with SVC

```
svc = SVC(random_state=1) #use support vector
    classifier setting up random state as 1
svc.get_params() # to get the all parameters \\
```

Listing 1. use of SVC with random state 1

B. GridSearchCV

```
svc_grid = {
    'kernel': ['poly', 'rbf', 'sigmoid', 'linear'],
    'C': np.linspace(.1, .9, 6)
}
```

Listing 2. To obtain best kernel

C. GridSearch estimator

```
cv = StratifiedKFold(n_splits=4, shuffle=True,
    random_state=1) #applying gridsearch method for
    getting the best parameters to apply on model
grid_search_estimator = GridSearchCV(svc, svc_grid,
    scoring='accuracy', cv=cv, n_jobs=-1)
grid_search_estimator.fit(X, y)
```

Listing 3. GridSearch for getting best parameters

D. Predict method of GridSearchCV

```

predicted = grid_search_estimator.best_estimator_.
    fit(X_train, y_train).predict(X_test) #Call
    predict on the estimator with the best found
    parameters.
```

Listing 4. Predict is called on estimator