What is Relevant in a Text Document a Machine Learning Based Approach

by

Abdullah Al Mahmud
17301033
Jannat-E-Noor
17101021
Sadman Alam Reshad
17101403
Syed Nafis Fuad
17101250

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Abdullah Al Mahmud	Jannat - E - Noor
Abdullah Al Mahmud 17301033	Jannat-E-Noor 17101021
Sadman Alam Reshad	Syed Nafis Fuad
Sadman Alam Reshad 17101403	Syed Nafis Fuad 17101250

Approval

The thesis/project titled "What is Relevant in a Text Document a Machine Learning Based Approach" submitted by

1. Abdullah Al Mahmud (17301033)
2. Jannat-E-Noor (17101021)
3. Sadman Alam Reshad (17101403)
4. Syed Nafis Fuad (17101250)

Of Summer, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 6, 2021.

Examining Committee:

Supervisor:
(Member)

[illegible]

Amitabha Chakrabarty, Ph.D
Associate Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, Ph.D
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

[illegible]

Sadia Hamid Kazi, Ph.D
Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Text Documents often contain valuable data. But not all data is relevant. That is why extracting relevant data from text documents is an essential task. Extracting relevant data from text documents refers to the study of classifying text documents into such groups that describe the contents of documents. There are many methods to find out relevant data from a cluster of text or a text document. Classifying extensive textual data helps to organize the records better, make the search easier and relevant and simplify navigation. That makes this task still an open research issue. This paper uses three techniques of classifying text documents: convolution neural networks (CNN) with deep learning, Gaussian Naïve Bayes and support vector machines (SVM). With these three algorithms, the text we want to classify goes through three layers of checks. So, it gives us more reliability.

Keywords: CNN; SVM; Gaussian Naïve Bayes; text classification

Acknowledgement

This research has been supervised by Dr. Amitabha Chakrabarty, Associate Professor, Department of Computer Science and Engineering, BRAC University. We are thankful to all the faculty members of BRAC University who have motivated and inspired us throughout the entire undergraduate program.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN	Convolutional Neural Network
GNB	Gaussian Naïve Bayes
IDF	Inverse Document Frequency
LDA	Linear Discriminant Analysis
LRP	Layer wise Relevance Propagation
LSI	Latent Semantic Indexing
ML	Machine Learning
SFS	Sequential Feature Selection
SVM	Support Vector Machine
TF	Term Frequency

Chapter 1

Introduction

The measure of textual information is expanding largely each day. So, the need for efficient handling of these data is immense. Recognizing significant content from text documents has additionally become vital. So, importance recognition of data is one of the significant characteristics of academic study to recognize apposite data from various sources. If we can use the composition of any document, we can differentiate the related subjects related to that particular document. A few documents have unsurprising structures in which the different parts ought to be anything but difficult to recognize, while a few documents are essential and smooth.

Text documents can be classified by different concepts. For example, classification based on semantic meaning and classification based on sentimental meaning. This paper aims to classify text documents using convolution neural networks (CNN), Gaussian Naïve Bayes, and support vector machines (SVM).

Text classification is very useful in many fields, including academia and industry. In our paper, we aim to find more useful use of the techniques mentioned above to classify text documents. Our main objective is to acquire as most efficient results possible and dive even deeper.

1.1 Problem Statement

Text classification, overall, is a rising field of study. Fields, for example, Marketing, Product Management, Academia, and Governance, are now utilizing the way toward analyzing and extracting data from textual information. But with the ever-growing world of information, finding relevant data from text documents is a very difficult task. Especially when it comes to search for millions of documents. Despite being such an important issue, there are still open research areas in this field.

This paper uses three techniques for the classification of text documents CNN, SVM, and Gaussian Naive Bayes. Our goal is to classify text documents that explain the contents of the documents. Herewith Gaussian Naive Bayes and SVM, we focus on classification based on word count, and with CNN, we focus on classification based on semantic meaning.

1.2 Research Objective

This research aims to find more systematic uses of Machine Learning to find relevant information from text documents. We have quite a few options to choose from when it comes to deep learning. The ML model of a trained CNN or Convolutional Neural Network, the Bag of Words SVM classifier model, and Gaussian Naive Bayes. Our proposed model categorizes the text by using these methods. By the end of our research, we hope to achieve a more constructive system for filtering relevant text. This research will certainly allow us to understand text classification in detail and further will help us in finding out solutions to the same type of problems and also updating our model.

Multi-layered data in hierarchical order presents different patterns in them, which can be implemented in deep learning. We are proposing to firstly identify the specific words that help to categorize the document into separate categories or genres with which the text is associated.

Chapter 2

Related Work

In this chapter, we have included all previous work related to our topic of text classification done by previous researchers.

E. Agichtein, L. Gravano, compared snowball with DIPRE, which needs less training data. In this paper, tuples were extracted using Snowball, and not all relevant information was extracted. Given some tuple, Snowball inspects the text that connects 'o' and 'l' to create a pattern [1].

R. W. White discussed the evaluation of two techniques that are related to a web search. The first one is the summarization technique which is after a search engine has found results, the top 30 websites are summarized based on words contain, their position, proportion of query terms. And the second one is implicit feedback, where context information is used; the idea of context is based on whether the user spends more time to read relevant material than less time to read less relevant material [2].

In the paper of L. M. Abualigah, A. T. Khader, E. S. Hanandeh, two basic KH algorithms are used (KHA) to find the solution to text document clustering problem. The first one is KHA, where two genetic operators are also used. And the other one is without genetic operators. They proposed three unique versions of the hybrid KH algorithm (HKHAs), which are HKHA1, HKHA2, and HKHA3. Furthermore, they also proposed a combination of objective functions [3].

A. Dhar, N. Dash, and K. Roy, in their paper, used cosine similarity and Euclidean distance to measure the vector space model. This model was based on the TF-IDF feature. Bangla text documents were taken as input which was tokenized. Then a vector space model was created, and cosine similarity and Euclidean distance were used separately [4].

Pattern deploying and pattern evolving techniques have been used to retrieve relevant documents [5]. Bucket models for mining text documents are a well-known method. A bucket is created using vectors, where each vector is a binary vector representation of a document.

It is assuming that positive documents are drawn from a solitary fundamental distribution, a compact support help to bind them together across all buckets. Negatives show a huge variety. Mining each container to track down the frequent item sets that fulfill a given support level. Each subsequent item set is a bunch of words. The consequence of this interaction is an assortment of sets of item sets, recovering the archives that help the item sets that are frequent in buckets [6].

K. Torkkola, in his paper, pointed out insufficiency in class inequality of two famous methods, LSI and SFS, according to some relevant criterion. He suggested the transformation of features on the basis of LDA. He suggested a systematic dimension reduction step by using LSI. [7].

F. Horn. L. Arras, G. Montavon, K. R. Muller, and W. Samek proposed a model that will find out relevant words from a document and also visualize them in word clouds. This model compared three methods of bringing out relevant words. This model used raw TF-IDF features, LRP, which breaks down the classification score. The relevancy score was computed by comparing the frequency of one class compared to other classes [8].

Deep learning has been another method to predict document classification. A proposed approach uses CNN with deep learning to predict classes of text. In addition, MNB was used [9].

In another paper by L. Arras, F. Horn, K. R. Muller used two machine learning models which were word-based, a CNN and an SVM classifier. This model used the LRP method. Reason behind it is to decompose the predictions of these models. And a CNN model. This CNN model had already been trained. So that it could map documents accurately to their actual category. Four steps were done to compute an input representation, forward- propagate the input representation through the CNN, backward-propagate the output through the network where LRP was used, pool the relevance scores associated to each input variable of the network [10].

Vector space model used by T. Xia and Y. Du in their paper used VSM text classification. They used VSM to represent documents with vectors. A collection of documents was indexed rather than individual documents to make it easier to categorize the documents. Terms in document titles and not in the titles were given different treatments [11].

Chapter 3

Data Description

Thousands of text data is available for research purpose. For our work, we chose "20 Newsgroup data". It is a collection of news articles of 20 different categories. For our work, we used ten categories. The categories that we used are as follows:
Atheism, Religion (miscellaneous), Computer Graphics, Space (Science), Microsoft Windows (miscellaneous), For sale (miscellaneous), Automobile vehicles, Sports (Baseball), Electronics Politics. 20 Newsgroup data is commonly used data that is used most often for text-related research. Here is the distribution of all different categories of this dataset:

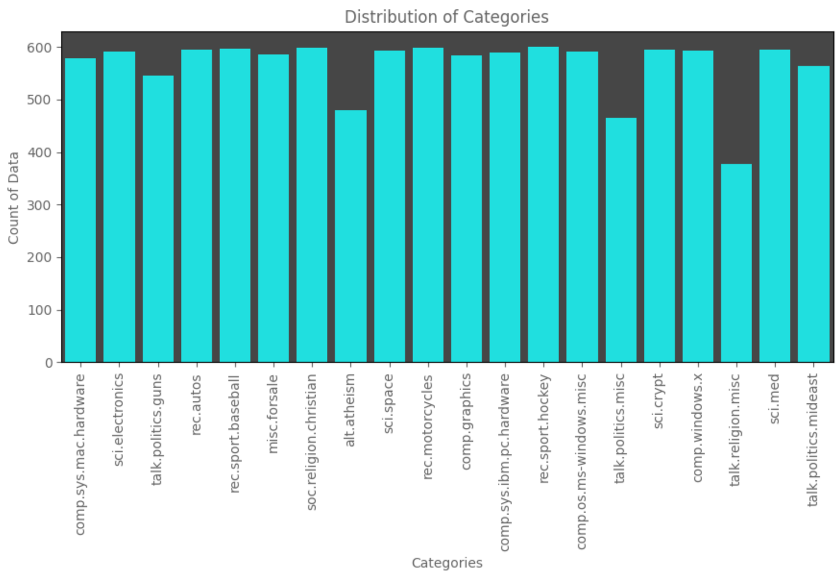


Figure 3.1: Distribution of Categories

As we can see in Figure 5.1, there are 20 different categories in this dataset. Almost all the categories have around 600 data. Again, let us take a look at this data after we have turn the text data into vectors.

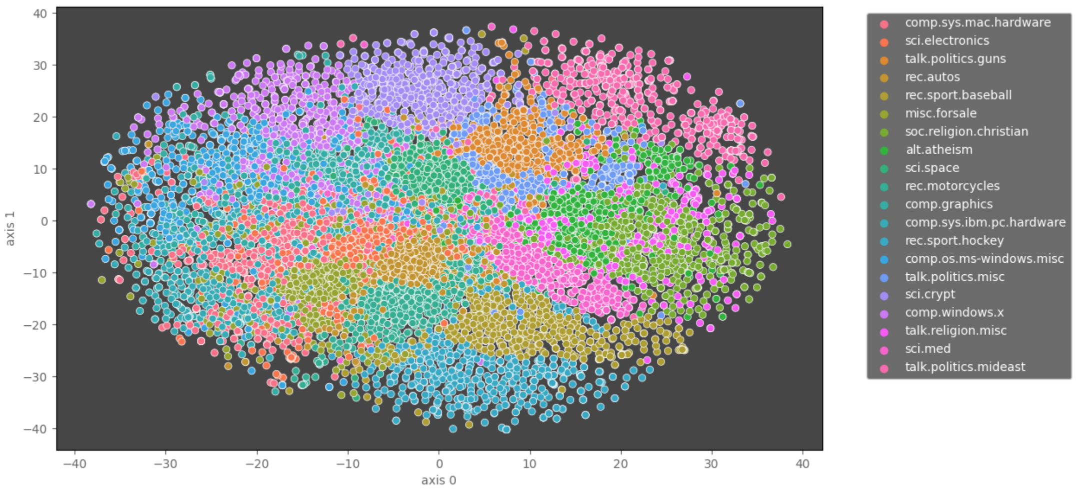


Figure 3.2: Vectorized form of Visualization

3.1 Data Preprocessing

In our model data preprocessing involves five steps.

1. Load sentences from raw data files.

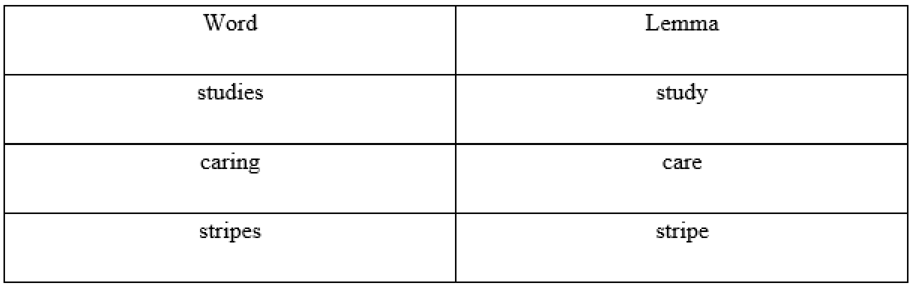
2. Removing stop words. Stop words are word that usually have no value in terms of text classification. These word may occur multiple times in a document but these words do not help to find any meaning. These words are used in a language to maintain the rules of grammar. But in text classification these words are not helpful. So by removing these words we can make the documents easier to process. Example of stop word removal:

You were too busy so I came back				
You	busy	I	came	back

3. Stemming and lemmatization. Stemming means transforming a word to its root form. For example, if we apply stemming on the word 'running', it will turn into 'run'. Again, applying stemming on the word 'cats' turns it into 'cat'. Example of stemming:

Word	Stemmed Word
trouble	troubl
dogs	dog
programming	program

Then comes lemmatizing words. Lemmatizing word means doing some processing with the help of vocabulary and morphological analysis or words. After lemmatizing a word, the token that we get is called a lemma. Some examples of lemmatization are given below:



Word	Lemma
studies	study
caring	care
stripes	stripe

4. Pad each sentence to the maximum sentence length. Padding is very important for neural network models. Padding means adding some value at the end or at the start of the sentence after it has been converted into vector form. Usually all the sentences are padded in such a way so that length of all sentences become same. So that is why each sentence is padded to the maximum sentence length.

5. The final step is creating a vocabulary index and after that mapping each word to an integer between 0 and n (where n is the vocabulary size). As a result, sentence is now a vector of integers.

Chapter 4

Methodology

The proposed model in this paper consists of three methods. CNN, SVM and Gaussian Naïve bayes. The main objective of this model is to classify text from text documents. Let us take a look at those three methods one by one.

4.1 Convolution Neural Network

The neural network model that we propose looks roughly as follows:

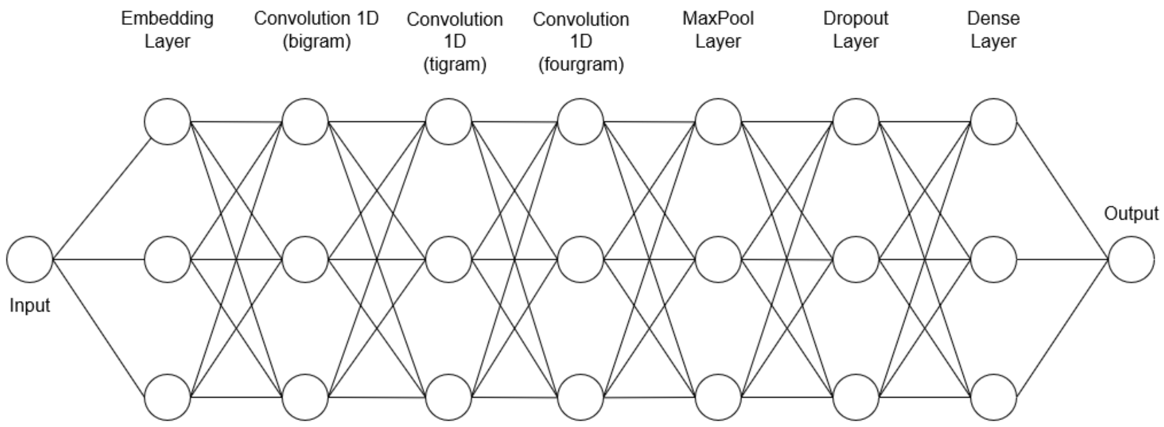
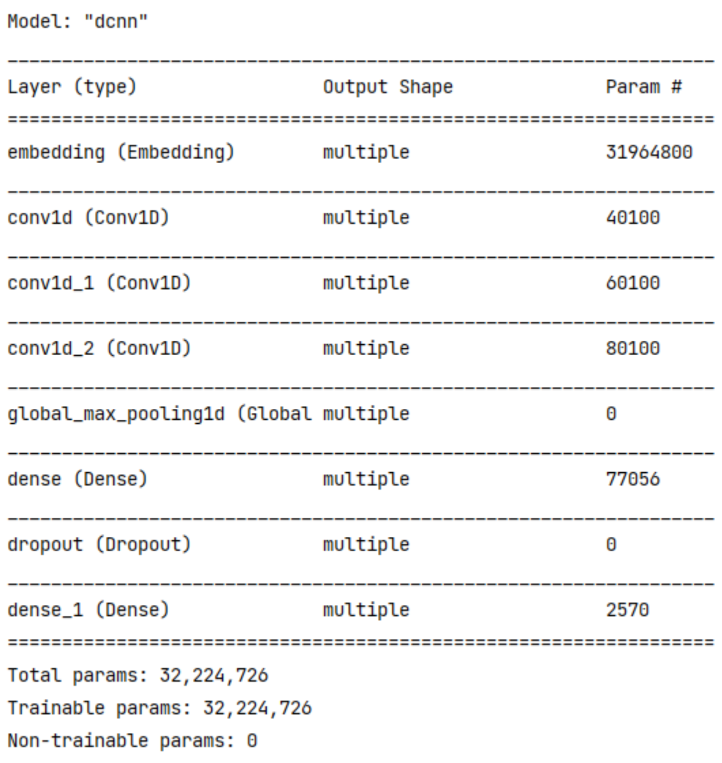


Figure 4.1: CNN

Now let us take a look of how each layer is helping to classify text data. The summary of this model looks as following:

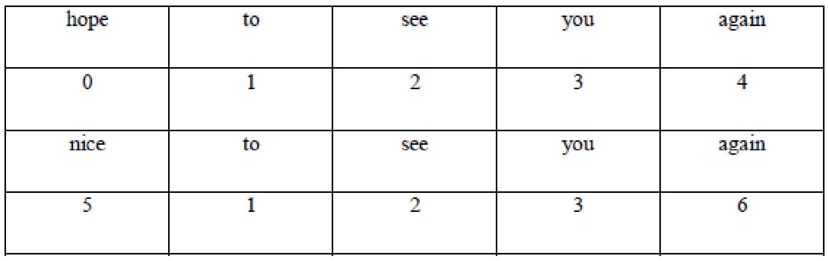


```
Model: "dcnn"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        multiple                  31964800  
_________________________________________________________________
conv1d (Conv1D)              multiple                  40100     
_________________________________________________________________
conv1d_1 (Conv1D)            multiple                  60100     
_________________________________________________________________
conv1d_2 (Conv1D)            multiple                  80100     
_________________________________________________________________
global_max_pooling1d (Global multiple                  0         
_________________________________________________________________
dense (Dense)                multiple                  77056     
_________________________________________________________________
dropout (Dropout)            multiple                  0         
_________________________________________________________________
dense_1 (Dense)              multiple                  2570      
=================================================================
Total params: 32,224,726
Trainable params: 32,224,726
Non-trainable params: 0
```

Figure 4.2: Summary of CNN

4.1.1 Embedding Layer

The embedding layer converts a word into a word vector and uses these word vectors to pass these to the next layer in the model. The word vector can have any dimensional space. In this model, we have used 128 as the dimension of a single word vector. And the number of vectors created is equal to the vocabulary size of our training data which is 159824. For an explanation of how embedding is done in this layer, let us take two sentences as an example. The first one is “Hope to see you soon,” and the other one is “Nice to see you again.” Now first, these sentences are encoded by assigning a unique integer number to each word.



hope	to	see	you	again
0	1	2	3	4
nice	to	see	you	again
5	1	2	3	6

Suppose we want the size of vectors to be 2. Then the above encoded sentences will look like these:

Index	Embedding
0	[1.2, 3.1]
1	[0.1, 4.2]
2	[1.0, 4.1]
3	[0.3, 2.1]
4	[2.2, 1.4]
5	[0.7, 1.7]
6	[1.4, 2.0]

So finally the trained data will have a vector look like following:

[[1.2, 3.1], [0.1, 4.2], [1.0, 4.1], [0.3, 2.1], [2.2, 1.4], [0.7, 1.7], [1.4, 2.0]]
This vector is then passed to the next layer.

4.1.2 Convolution 1D Layer

A convolution 1D layers creates a convolution kernel. This kernel is convolved with the layer input over a single semantic dimension. This layer produces a tensor of outputs. So this is basically a matrix of dimension (vocabulary size * dimension of word vector). According to this, the model that we have used will have a convolution 1D of size (159824 * 128). Let us take a look at an example.

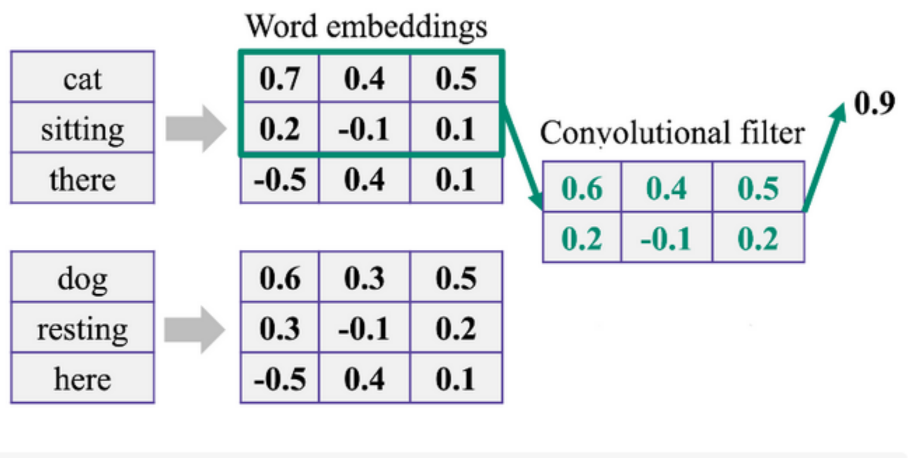


Figure 4.3: Word embedding

In the above figure size of the kernel is 2. The kernel iterates through the whole matrix and keeps generating a convolution filter or tensor. As the kernel in the figure uses a kernel of size 2, this is called a convolution 1D bigram. The difference among convolution 1D bigram, trigram, and fourgram is the kernel size used. In bigram kernel size is 2, in trigram kernel size is three, and in fourgram, kernel size is 4.

4.1.3 MaxPool Layer

In this model, we have used a one-dimensional MaxPool layer. A one-dimensional MaxPool layer reduces the size of data, the number of parameters, amount of computation, and also controls overfitting. A one-dimensional max pool block moves a window over the input data with a specific stride. While doing this, it computes the maximum value in each window. This layer helps the convolution layer to retrieve information from a bigger portion of the original vector.

4.1.4 Drop Out Layer

When there are many layers in a neural network, there are many weights and many bias parameters. This leads to overfitting problems. A way to fix this problem is to add a dropout layer in the network. A dropout rate is passed to this layer. What this layer does is deactivates some neurons in a layer on the basis of this value. In each iteration, this is done randomly. This means neurons are deactivated randomly based on the probabilistic value of the dropout rate. As a result of some neurons being deactivated, the chances of overfitting the dataset are decreased.

4.1.5 Dense Layer

A dense layer connects each input to each output. Dense layer uses the following operation to find out the output:

$$output = activation(dot(input, kernel) + bias) \tag{4.1}$$

here element wise activation is performed by activation and the kernel is a weight matrix. Kernel is created by the layer. Bias is a vector, also created by the layer. The output generated by this layer is a vector. This layer basically changes the dimension of the input vector.

4.2 Support Vector Machine

Support vector machine is a machine learning model that is based on the structural risk reduction principle from computational learning theory [12]. Structural risk minimization is basically the idea of finding a hypothesis h for which a lower true error can be guaranteed. Here, h the probability of making a wrong assumption on an unseen and randomly selected text example.

For SVM, we have used the TF-IDF method to vectorize words. TF-IDF evaluates the relevancy of a word in a document. It is calculated by multiplying two metrics. These two metrics are TF and IDF. There are many ways of measuring TF. One of them is counting the number of times a word appears in a document. On the other hand, IDF means how much relevant or irrelevant a word is in the entire document. An IDF value of a word that is close to 0 describes that the word is more relevant in the document [13].

4.3 Gaussian Naïve Bayes

This is a classification technique that uses the Bayes' Theorem. According to this theorem, a classifier makes the assumption that the existence of a particular feature in a class is not related to the presence of any other class. This theorem calculates posterior probability.

$$\begin{gathered} P(c \mid x)=\tfrac{P(x \mid c) P(c)}{P(x)} \\ P(c \mid X)=P\left(x_1 \mid c\right) \times P\left(x_2 \mid c\right) \times \ldots \times P\left(x_n \mid c\right) \times P(c) \end{gathered} \tag{4.2}$$

In the above equation, $P(c \mid x)$ is the posterior probability of class c given that predictor is x, P(c) is the prior probability of class c, P(x) is the prior probability of predictor x, and finally, $P(x \mid c)$ is the probability of prediction given class. Gaussian Naïve Bayes follows a normal distribution, and it supports continuous data. This means Gaussian Naïve Bayes makes an assumption that the continuous values related to each class are distributed according to normal distribution. Gaussian Naïve Bayes is calculated with the following equation:

$$P\left(x_i \mid y\right)=\frac{1}{\sqrt{2 \pi \sigma_y^2}} \exp \left(-\frac{\left(x_i-\mu_y\right)^2}{2 \sigma_y^2}\right) \tag{4.3}$$

The model we have proposed uses above mentioned three algorithms to classify text data. The working of the model can be represented by the following flow chart:

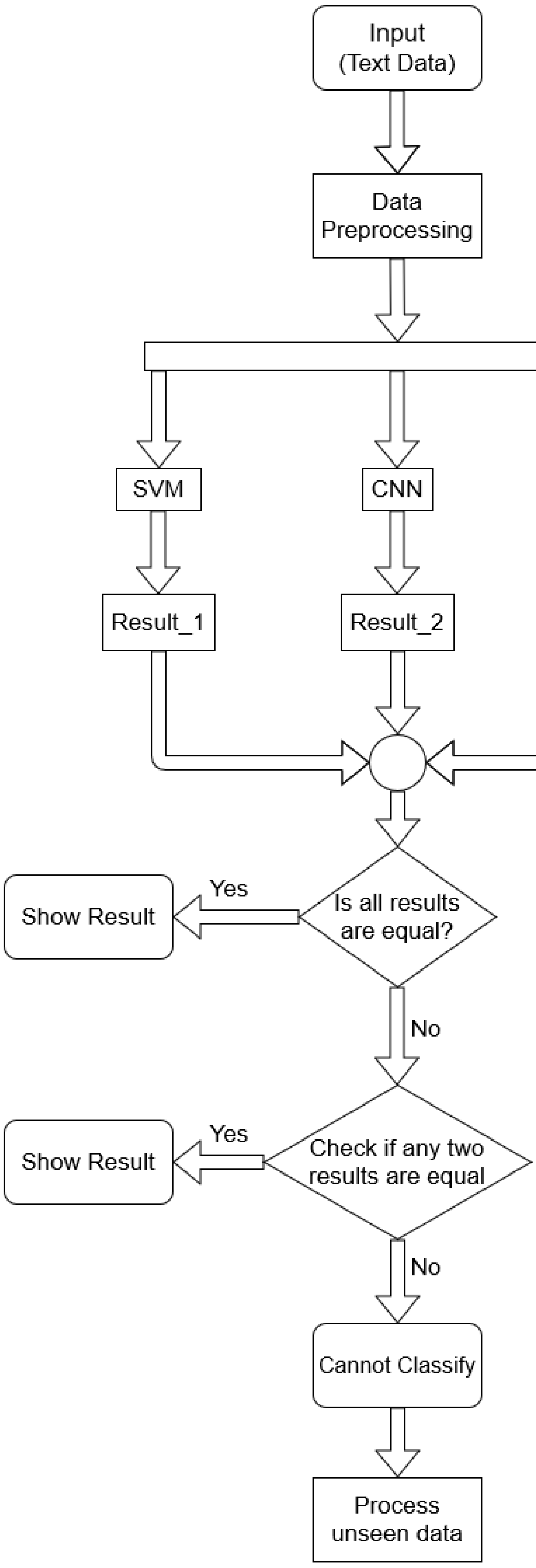


Figure 4.4: Flowchart of our Model

Chapter 5

Result Analysis

In this chapter, we analyze the results extracted from training and testing the models. We trained three algorithms separately as in our model they classify input independent of each other.
Support Vector Machine
With SVM we achieved,
Train accuracy score: 99.93%
Test accuracy score: 89.46%

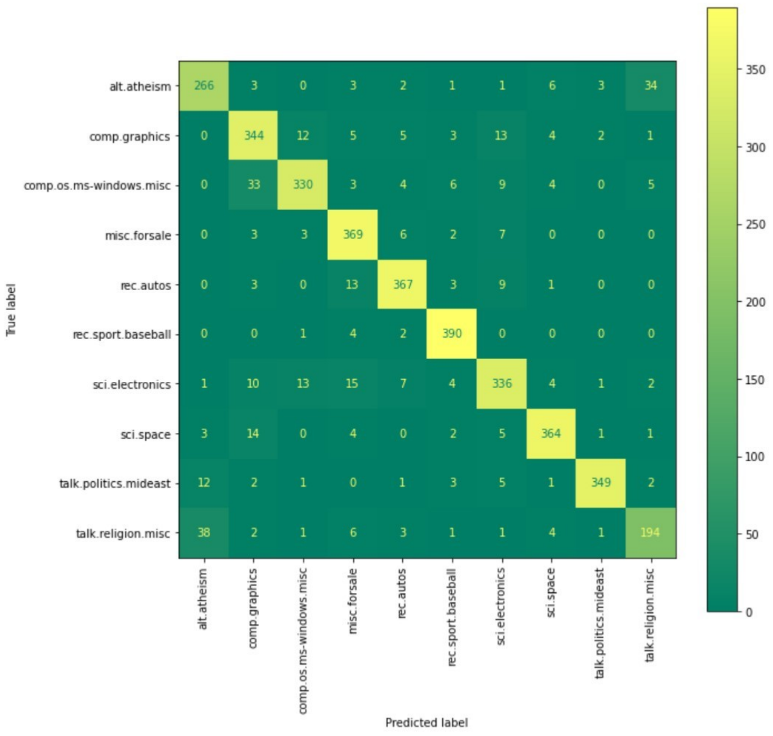
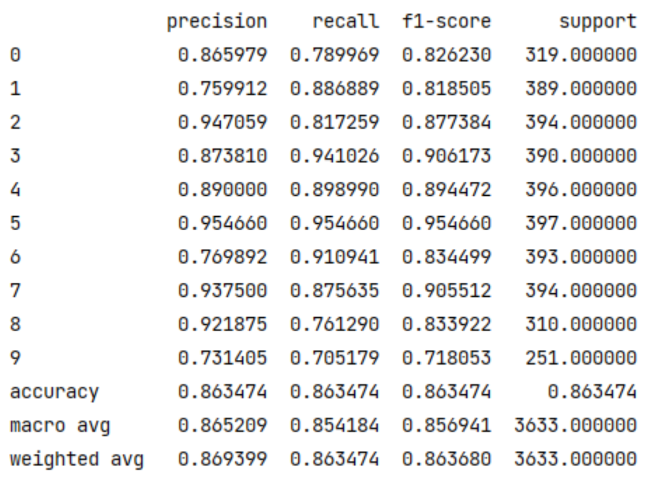


Figure 5.1: True values vs Predicted values

Here are the results of precision and f1-score:



```
              precision    recall  f1-score      support
0              0.865979  0.789969  0.826230   319.000000
1              0.759912  0.886889  0.818505   389.000000
2              0.947059  0.817259  0.877384   394.000000
3              0.873810  0.941026  0.906173   390.000000
4              0.890000  0.898990  0.894472   396.000000
5              0.954660  0.954660  0.954660   397.000000
6              0.769892  0.910941  0.834499   393.000000
7              0.937500  0.875635  0.905512   394.000000
8              0.921875  0.761290  0.833922   310.000000
9              0.731405  0.705179  0.718053   251.000000
accuracy       0.863474  0.863474  0.863474     0.863474
macro avg      0.865209  0.854184  0.856941  3633.000000
weighted avg   0.869399  0.863474  0.863680  3633.000000
```

Figure 5.2: Results of precision and f1-score

Gaussian Naïve Bayes With GNB we achieved,
Train accuracy score: 99.77%
Test accuracy score: 75.62%

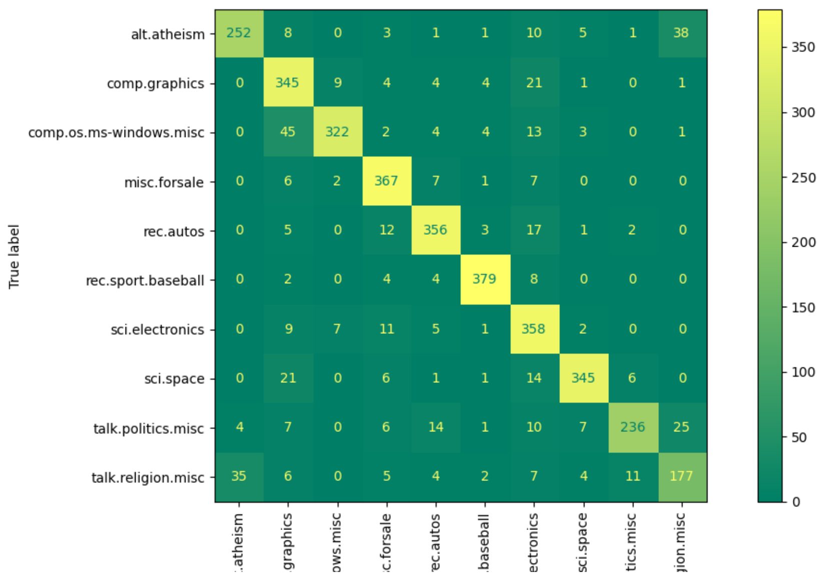
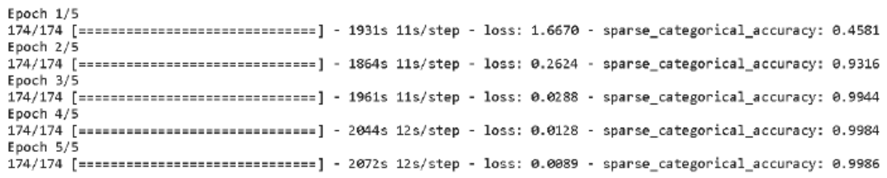


Figure 5.3: True values vs Predicted values

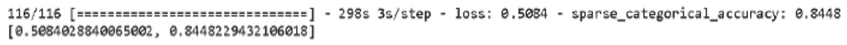
Convolutional Neural Network
With CNN we received,



```
Epoch 1/5
174/174 [==============================] - 1931s 11s/step - loss: 1.6670 - sparse_categorical_accuracy: 0.4581
Epoch 2/5
174/174 [==============================] - 1864s 11s/step - loss: 0.2624 - sparse_categorical_accuracy: 0.9316
Epoch 3/5
174/174 [==============================] - 1961s 11s/step - loss: 0.0288 - sparse_categorical_accuracy: 0.9944
Epoch 4/5
174/174 [==============================] - 2044s 12s/step - loss: 0.0128 - sparse_categorical_accuracy: 0.9984
Epoch 5/5
174/174 [==============================] - 2072s 12s/step - loss: 0.0089 - sparse_categorical_accuracy: 0.9986
```

Figure 5.4: Train Accuracy of CNN

Train accuracy score: 99.86%

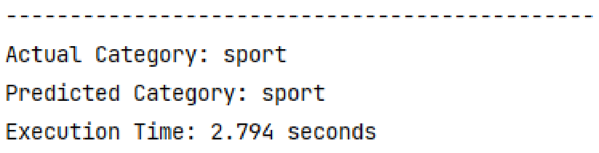


```
116/116 [==============================] - 298s 3s/step - loss: 0.5084 - sparse_categorical_accuracy: 0.8448
[0.5084028840065002, 0.8448229432106018]
```

Figure 5.5: Test Accuracy of CNN

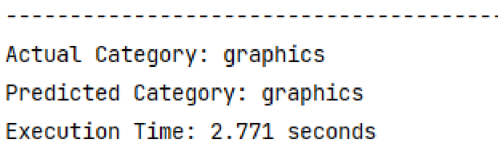
Test accuracy score: 84.48%

So, the average accuracy of these three algorithms is 83.19% which is overall the accuracy of our model. After building our model we used some random data from the test data set as input and here are the results:



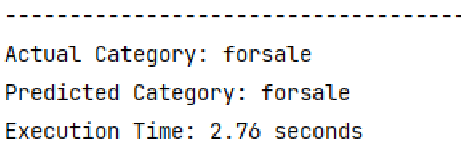
```
------------------------------------------------
Actual Category: sport
Predicted Category: sport
Execution Time: 2.794 seconds
```

Figure 5.6: Prediction 1



```
-----------------------------------------
Actual Category: graphics
Predicted Category: graphics
Execution Time: 2.771 seconds
```

Figure 5.7: Prediction 2



```
--------------------------------------
Actual Category: forsale
Predicted Category: forsale
Execution Time: 2.76 seconds
```

Figure 5.8: Prediction 3

Training Result Graph:

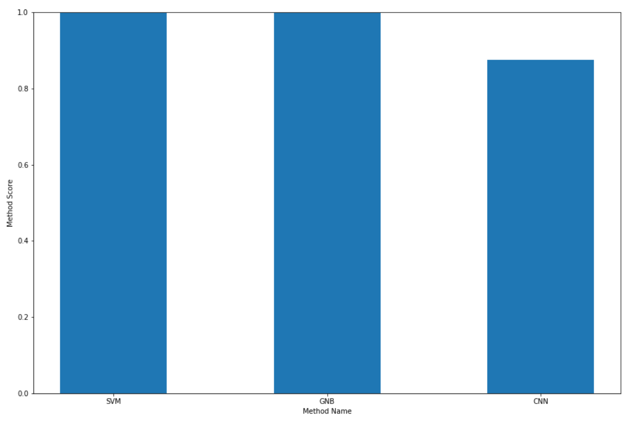


Figure 5.9: Training Result Graph

Test Result Graph:

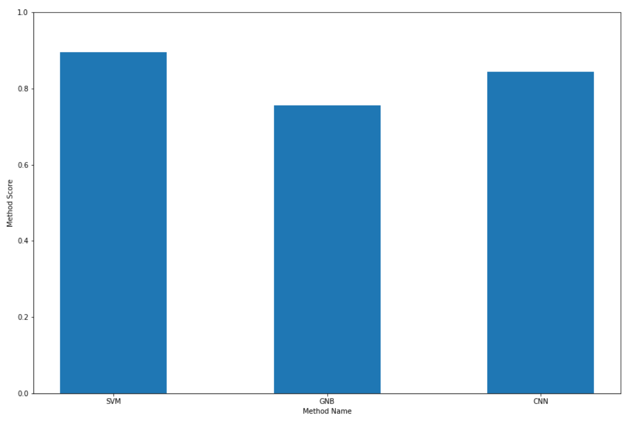


Figure 5.10: Test Result Graph

Chapter 6

Future Work

There are still quite a few spaces for improvement in our research. According to the flowchart of figure 6.4 (flowchart of our model), chapter 6, our model will not be able to classify the text data if all the algorithms give different results. Again, if two models give the same result and both are wrong, then our model will not be able to classify correctly. Although our model provides reliability, there are still some flaws present in the model. So we would like to improve on these flaws by using better approaches. We would also like to make the process of predicting faster, so it becomes more usable.

Chapter 7

Conclusion

The main purpose of this paper is to develop a model to classify any text data. We used 20 newsgroup datasets to train our model. Our model consists of three algorithms. Convolutional Neural Network, Support Vector Machine, and Gaussian Naïve Bayes. These three algorithms predict results independently of each other after that; all the results are compared to find out the correct classification. This provides much reliability to the outcome. But there are still some flaws in this model. So there are places for improvement. We would like to use the knowledge that we gained while conducting this research to improve this model and make this model more reliable, efficient, accurate, and usable.

Bibliography

[1] Eugene Agichtein and Luis Gravano. “Snowball: Extracting relations from large plain-text collections”. In: *Proceedings of the fifth ACM conference on Digital libraries*. 2000, pp. 85–94.

[2] Ryen W White, Ian Ruthven, and Joemon M Jose. “Finding relevant documents using top ranking sentences: an evaluation of two alternative schemes”. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002, pp. 57–64.

[3] Laith Mohammad Abualigah, Ahamad Tajudin Khader, and Essam Said Hanandeh. “A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis”. In: *Engineering Applications of Artificial Intelligence* 73 (2018), pp. 111–125.

[4] Ankita Dhar, NiladriSekhar Dash, and Kaushik Roy. “Classification of text documents through distance measurement: An experiment with multi-domain Bangla text documents”. In: *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall)*. IEEE. 2017, pp. 1–6.

[5] Shivani D Gupta and BP Vasgi. “Implementation of pattern discovery to retrieve relevant document using text mining”. In: *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE. 2015, pp. 327–332.

[6] Daniel Barbará, Carlotta Domeniconi, and Ning Kang. “Mining relevant text from unlabelled documents”. In: *Third IEEE International Conference on Data Mining*. IEEE. 2003, pp. 489–492.

[7] Kari Torkkola. “Discriminative features for text document classification”. In: *Formal Pattern Analysis & Applications* 6.4 (2004), pp. 301–308.

[8] Franziska Horn et al. “Exploring text datasets by visualizing relevant words”. In: *arXiv preprint arXiv:1707.05261* (2017).

[9] P Parvathi and TS Jyothis. “Identifying Relevant Text from Text Document Using Deep Learning”. In: *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*. IEEE. 2018, pp. 1–4.

[10] Leila Arras et al. “” What is relevant in a text document?”: An interpretable machine learning approach”. In: *PloS one* 12.8 (2017), e0181142.

[11] Tian Xia and Yi Du. “Improve VSM text classification by title vector based document representation method”. In: *2011 6th International Conference on Computer Science & Education (ICCSE)*. IEEE. 2011, pp. 210–213.

[12] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[13] B Stecanella. *What is tf-idf*. 2019.

Appendix A

Appendix

A.1 Feedback

In this section, we will add the feedback we had received from the panel members at the defense session. The criticisms were about:

- Improper Wording in Abstract.
- Lacking Chapter Description.
  We have accepted the feedback and improved our paper according to the criticisms by reviewing our choice for words in the Abstract and adding short descriptions to the beginning of each chapter.

We have accepted the feedback and improved our paper according to the criticisms by reviewing our choice for words in the Abstract and adding short descriptions to the beginning of each chapter.