Analyzing 20 Newsgroups Data Set

Sai Vineeth Doddala
Department of Computer Science
University of California, Irvine
sdoddala@uci.edu

Vignatha Yenugutala
Department of Computer Science
University of California, Irvine
vyenugut@uci.edu

Manikanta Loya
Department of Computer Science
University of California, Irvine
manikanl@uci.edu

Abstract

In this project, we aim to analyze the 20 Newsgroups data set with different models and test their performances. Text data is encoded into word vectors using TF-IDF vectorizer or Glove embeddings and these are used as inputs for classification tasks. Traditional algorithms like KNN and Naive Bayes performed poorly on this classification task as these are simple models and fail to extract complex patterns within the data and also do not understand the context. We implemented powerful models like Support Vector Machines, Neural Networks, Hierarchical Attention Network and Text CNN on word vectors for this classification. After tuning the hyper-parameters, we compared and analysed the results of these models on the data sets. We conclude that SVM and TextCNN models classified the text better than the other models.

1 Introduction

Text Classification (TC) is the classification of documents concerning a set of one or more pre-existing categories. TC is a hard and very useful operation frequently applied to assign subject categories to documents, to route and filter texts, or as a part of natural language processing systems. In this project, we explored different machine learning models and evaluated their performance for text classification tasks using 20 Newsgroups data-set.

We cannot directly use text data as inputs to our ML models. So, we used pre-processing techniques like Stop words removal, stemming, lemmatization to clean the dataset. And then we converted the resulting words into vectors, which can be used as inputs to the model, using techniques such as Count Vectorizer, TF-IDF, and Glove embeddings.

To perform the classification task we trained models like KNN and Neural Networks. These models performed poorly on text classification tasks as they fail to understand the sequential occurrence relationship between words. We also used SVM which performed better than previously mentioned classifiers. To incorporate contextual understanding into decision making we used sequential models like Hierarchical Attention Network(HAN) and spatial models like TextCNN. We found that SVM, TextCNN and HAN perform better than simpler models.

2 Data cleaning and preparation

The data set consists of texts from 20 classes which are, alt.atheism, comp.graphics,comp.os.ms-windows, alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware,

Preprint. Under review.

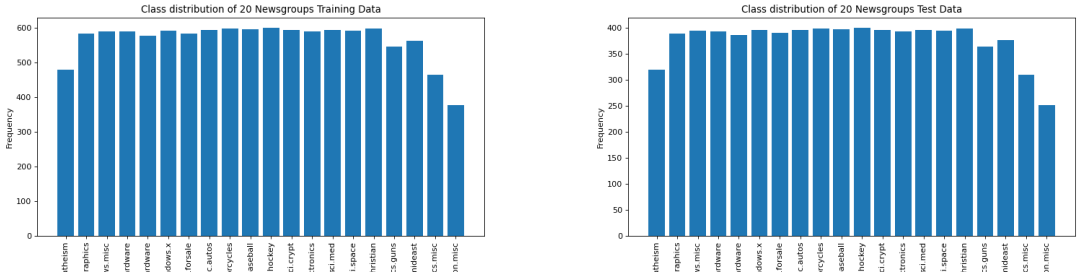

Figure 1: Distribution of data across the classes in training and testing data sets

comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc.
The data set is divided into training and test data sets. The training data set contains 11314 data points while test data set contains 7532 data points. The plots of distribution of classes in each of the datasets can be found in Figure 1.

An example of a datapoint in the data set is as follows:
"A fair number of brave souls who upgraded their SI clock oscillator have\nshared their experiences for this poll. Please send a brief message detailing\nyour experiences with the procedure. Top speed attained, CPU rated speed,\nadd on cards and adapters, heat sinks, hour of usage per day, floppy disk\nfunctionality with 800 and 1.4 m floppies are especially requested.\n\ nI will be summarizing in the next two days, so please add to the network\nknowledge base if you have done the clock upgrade and haven't answered this\npoll. Thanks."

2.1 Noise Removal and Lower-casing

We need to clean the data set of escape characters, special characters, numbers, line spaces etc. To maintain uniformity over the data we need to convert the text into either Lowercase or Uppercase characters. We have converted the entire text into lowercase letters.

After noise removal, the above example got refined as follows:
'a fair number of brave souls who upgraded their si clock oscillator have shared their experiences for this poll please send a brief message detailing your experiences with the procedure top speed attained cpu rated speed add on cards and adapters heat sinks hour of usage per day floppy disk functionality with and m floppies are especially requested i will be summarizing in the next two days so please add to the network knowledge base if you have done the clock upgrade and havent answered this poll thanks'

2.2 Stop Words

Words such as "is", "which", "the", "and", "of" make text data noisy, and that will reduce the efficiency of text data documents. Stop words typically point to the most widely-recognized words in a language.Stop word removal or stopping word aims to eliminate extremely common words by using a stop word list that has partial predictive worth for classification.

The above example got refined after removing stopwords as follows:
'fair number brave souls upgraded si clock oscillator shared experiences poll please send brief message detailing experiences procedure top speed attained cpu rated speed add cards adapters heat sinks hour usage per day floppy disk functionality floppies especially requested summarizing next two days please add network knowledge base done clock upgrade havent answered poll thanks'

2.3 Lemmatization and Stemming

In English, the words ”smoker” and ”smoking” have the same root stem ”smoke”. The main purpose of lemmatization is to cut words down to their root. In stemming, selected words are reduced to their word stem such as the word ”waiting”, ”waits”, and ”waited” would all to be reduced to single feature which is ”wait”. Stemming is usually done by removing any attached suffixes and prefixes (affixes) from index terms before the actual assignment of the term to the index.

The example gets refined as follows:
'fair number brave soul upgrad si clock oscil share experi poll pleas send brief messag detail experi procedur top speed attain cpu rate speed add card adapt heat sink hour usag per day floppi disk function floppi especi request summar next two day pleas add network knowledg base done clock upgrad havent answer poll thank'

2.4 Count Vectorizer

CountVectorizer is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we want to convert each word in each text into vectors. CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell in the matrix is the count of the word in that particular text sample.

2.5 TF-IDF Vectorizer

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a corpus is to a text. The TF-IDF value increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

2.6 Glove Embeddings

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

3 Models under study and Experiments

3.1 KNN

We thought of starting with a simple model and hence we tried K-nearest neighbors on this test data after following the pre-processing methods mentioned in the above section. We varied the number of neighbors and as we can see in Figure 2b, the highest accuracy is obtained with k=600, and the accuracy is reduced after that point. As kNN is a very simple model, it is difficult for the algorithm to produce good results on text data. As the final accuracy(62%) is not very good, we moved to other complex models.

3.2 Neural Networks

Next, we thought of using a complex and powerful model and hence we tried Neural Networks on this dataset. First, we performed basic pre-processing like removing alphanumeric characters, converting to lowercase, removing stop words, and lemmatization as explained before. Since a neural network can't work on text data directly, we converted text data into vectors using TF-IDF vectorizer. We first built a sequential model with one fully connected layer and a 'relu' activation function. We observed that the training accuracy is reaching 96% but the test accuracy is just reaching 68%. We thought of increasing the complexity by adding a few more layers and also tried a 'sigmoid' activation function. When we trained the model with three layers, we observed that the training accuracy is reaching 98% but the test accuracy is just reaching 69%. Looking at the generalization error, we can say that the model is over-fitting the data. So we tried a regularization method called dropout to reduce the

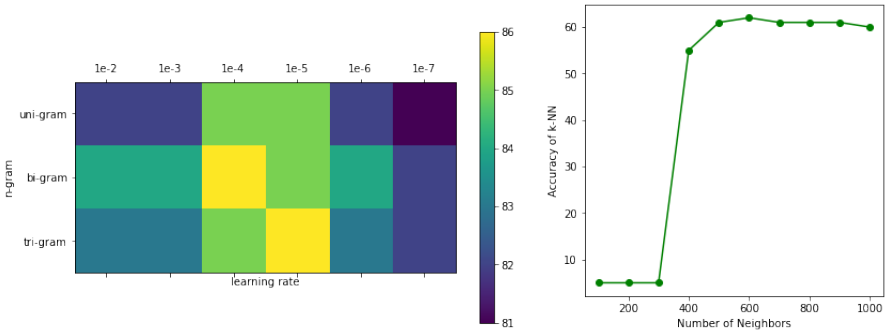

Figure 2: a) Accuracy of SVM model by varying learing rate and n-grams. b) Accuracy of kNN by varying number of neighbors

over-fitting. By trying various dropout numbers ranging from [0.2, 0.4] we could bring down the training accuracy to 97% but the test accuracy remained at 69%. With this experiment, we found that even complex models like Neural Networks couldn't perform well on this text data as they can't remember the context of the sentence.

3.3 Support Vector Machine(SVM)

Historically, SVM is known to perform well for text classification tasks. So we tried SVM on this dataset using both Count vectorizer and TF-IDF vectorizer. We used grid search and trained the model with hyperparameters such as n-grams, learning rate. With the TF-IDF vectorizer, we varied the learning rate in the range [1e-2 to 1e-7]. n-grams play a major role in text data, so we tried Uni-gram, Bi-gram, and Tri-gram for all the learning rates. We tried linear SVM and rbf kernel on this dataset. We observed that the highest accuracy(86%) on test data is obtained for learning rate 1e-4 and by using Bi-grams. Our experiment of changing learning rate and n-grams are captured in Figure 2a above. We tried a similar experiment using Count vectorizer and we observed the highest accuracy of 82% with learning rate 1e-2 and by using Bi-grams.

3.4 TextCNN

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features. Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP. CNN's learn to recognize patterns across space and it works well where detecting local and position-invariant patterns is important. In this project, we train a simple CNN with one layer of convolution on top of word vectors(embedding dimension is 128) obtained from an unsupervised neural Language model. We are using kernels of sizes 1,2,3,4, max-pooling layers, and a fully connected last layer. We used a dropout rate of 0.5 to reduce the over-fitting. We tried to vary the number of words that are considered from each document(400,1000 full data). We were able to achieve a test accuracy of 87.4% for full data by training for 10 epochs and with a batch size of 32.

3.5 Hierarchical Attention Network (HAN)

Here, we implemented a Hierarchical Attention Network that takes context into account. The idea here is to derive sentence meaning from the words and then derive the meaning of the document from those sentences. Here, we use the attention model so that the sentence vector can have more attention on "important" words. The attention model consists of two parts: Bidirectional RNN and Attention networks. While bidirectional RNN learns the meaning behind those sequences of words and returns vector corresponding to each word, the Attention network gets weights corresponding to each word vector using its own shallow neural network. Then it aggregates the representation of those words to form a sentence vector that embodies the sentence and the same procedure applies to sentence vectors so that the final vector embodies the gist of the whole document. Since it has two levels of attention model, therefore, it is called hierarchical attention networks.

Initially, we performed the pre-processing methods, converting to lowercase, removing stop

words, and lemmatization. Here, we are using the Glove embeddings to convert the word into vectors. The average number of words in a sentence was observed as 24 while the average number of sentences came about 11. Hence the important parameters for HAN that are the maximum number of features(number of unique tokens that should be included in the tokenized word index) was selected as 200000, the maximum number of sentences in one document was chosen as 40. The maximum number of words in each sentence was chosen as 50 whereas the embedding size of the embedding layer was chosen as 100. We observed 70781 unique tokens in the data and 29214 which is 41.27% of total words were observed to be absent in the word index built by the Keras tokenizer. Initially, the training accuracy was 97% but the test accuracy remained at 69%. To control this over-fitting, dropout regularization was done with values in [0.3,0.8]. The best results came with a dropout rate of 0.5 where the training accuracy came out as 94% and testing accuracy remained at 75%. We think that because of the high number of absent words in the word index, we were unable to hit higher accuracy with this model. The plots of accuracy and loss values of training and test values are mentioned in Figure 3.

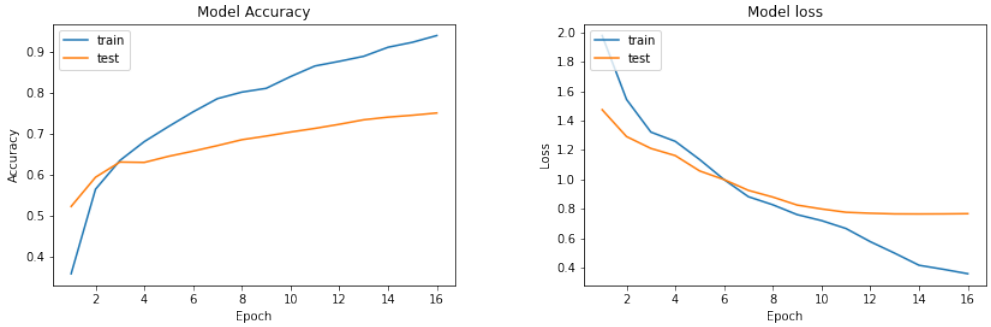

Figure 3: Plots of accuracy and loss values of training and test data sets as the models were trained.

4 Results and Conclusion

From the above analysis (Figure 1) in the Data cleaning section, the 20-Newsgroups is a well-balanced dataset. So measuring and comparing the accuracy of a model makes more sense.

Models	**Training Accuracy**	**Test Accuracy**
K nearest neighbors	66.76%	62.12%
Neural Networks	94.93%	69.52%
Hierarchical Attention Networks	94.10%	75.10%
SVM(with Count vectorizer)	100.0%	82.21%
SVM(with TF-IDF vectorizer)	100.0%	86.07%
TextCNN	96.87%	87.40%

Table 1: Performance of different models on the dataset

In Table 1, we have compared the accuracies of different algorithms on this data set. As expected, KNN performed poorly among all the models as it's a very simple model and because of a large number of feature dimensions. Neural networks performed relatively better than KNN as they can extract complex patterns within the dataset. But Neural Networks can get stuck at local minima and fail to converge to global minimums. SVM models can use higher dimensional feature spaces with the help of kernels like RBF functions and it converges to the global minimum. With help of Spatial and sequential models, we can classify the models better. Sequential Models like HAN are sensitive to weight initialization because of this they may be under-performing. We can achieve higher accuracy with the Text CNN model. We observed Text Classification tasks depend on feature extraction and understanding the context of word occurrences. In continuation to this work, we can use auto-encoder to train weight initialization for Attention models and the performance may improve.

5 Task Splitting

Our work was collaborative and was split evenly. Details of the individual tasks: Vineeth : Worked on the SVM and Neural Network models Vignatha: Hierarchical attention Models and K nearest neighbors Manikanta: Worked on Text CNN

6 References

[1] https://medium.com/analytics-vidhya/hierarchical-attention-networks-d220318cf87e

[2] https://humboldt-wi.github.io/blog/research/information_systems_1819/group5_han/

[3] https://www.cs.cmu.edu/~/hovy/papers/16HLT-hierarchical-attention-networks.pdf

[4] @misckim2014convolutional, title=Convolutional Neural Networks for Sentence Classification, author=Yoon Kim, year=2014, eprint=1408.5882, archivePrefix=arXiv, primaryClass=cs.CL