Multi-class Sentiment Analysis using Deep Learning

Tejas Wadiwala
Department of Computer Science
Lakehead University
Thunder Bay, Canada
twadiwal@lakeheadu.ca

***Abstract*—In this paper, the main aim is to implement a scalable and robust Convolutional Neural Network-based solution for the problem of text-based movie review multi-class sentiment analysis. The dataset on which multi-class sentiment analysis is performed is the rotten tomatoes movie reviews dataset. A model is created by defining the input layers and connected convolutional layers. There are hyperparameters also defined, which also play an essential role in the outcome. Various experiments have been performed, and I have got different outputs from the various experiments performed. These experiments have been performed by changing the layers of the model, also by changing the hyperparameters. The execution environment, i.e., the training and testing of the model, are done using Google Colab. The feature extraction can be done by using the vectorizers: Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Word2vec. The experimental analysis of the data is documented.**

***Index Terms*—Adam, Bag of Words (BoW), batch size, dataset, epochs, flatten, learning rate, Term Frequency-Inverse Document Frequency (TF-IDF)**

I. Introduction

Deep learning is a type of machine learning method which is based on artificial neural networks. Learning can be supervised, semi-supervised, or unsupervised. Deep learning uses multiple-layers to continuously extract higher-level features from raw input [1]. A convolutional neural network is a type of deep learning architecture that is used in natural language processing for sentiment analysis. Sentiment analysis is the classification of data into different categories using text analysis techniques. It can be a binary classification or a multi-class classification. The dataset that we are using has multiple categories; therefore, a multi-class classification problem. The categories in which the sentiments would be classified are negative, very negative, neutral, very positive, positive. This type of analysis is known as fine-grained sentiment analysis [2]. It is represented in the dataset in numerical form. The data is first preprocessed by removing the stopwords, using lemmatization to get the root word along-with its meaning, and removing the punctuations from the data. The output thus obtained is converted into vectors, which is known as vectorization. The data can be converted into vectors using the following: Bow, TF-IDF, and Word2vec. The vectorized data is then converted into a proper dimension. Keras is then used to build a convolutional neural network model that is used for training and testing the data.

II. Literature Review

A deep convolutional neural network-based pipeline is proposed for the diagnosis of Alzheimer's disease (AD) and its stages using magnetic resonance imaging (MRI) scans. Multi-class classification is done on the Alzheimer's disease Neuroimaging Initiative (ADNI) dataset, specifically a four-way classifier is implemented to classify AD, mild cognitive impairment (MCI), late mild cognitive impairment (LMCI), and healthy persons. The MRI scans were first preprocessed to get the gray matter images, which was then passed to the CNN network. The networks were trained and tested using deep GoogleNet and ResNet models. The authors suggested that the results from both the models outperformed all the other methods of literature in regard to multi-class classification. The authors were able to get a prediction accuracy of 98.8% [3].

Ankita Rane and Dr. Anand Kumar have compared customer feedback on 6 US Airlines using twitter data. The authors have worked on a dataset comprising of tweets of 6 major US Airlines and performed a multi-class sentiment analysis. The dataset was taken from Kaggle, and it consisted of 14640 tweets. For preprocessing, stemming was performed on the tweets to clean them and represented them as vectors using a deep learning concept (Doc2vec) to do a phrase-level analysis. The authors have suggested that the Doc2vec model is inspired by the Word2vec approach. Also, different classification techniques have been described - Decision Tree Classifier, Random Forest Classifier, Logistic Regression Classifier, Support Vector Machine Classifier, AdaBoost Classifier, and K-Nearest Neighbour Classifier. These are also used to compare the accuracies with each other. The highest accuracy which the authors achieved is 84.5%. The approach used in the classification can be used by airline companies to analyze feedback using the twitter data [4].

Bohang Chen et al., 2018, have proposed a deep sentiment representation model based on CNNs and long short-term memory recurrent neural network (LSTM). The model which they have proposed uses two layers of CNNs to capture partial features of the text. The model can capture more accurate partial features, which can capture the contextual information, after which the features fed to the LSTM. The main thing the authors have done is when the words have been learned by the system, which has been done using different convolution layers to obtain textual feature information, is then sent to a new convolution layer. Experimental analysis proposed on this model demonstrates that the model can achieve an accuracy upto 78.42% for multi class sentiment analysis [5].

III. Dataset

The dataset which is used for multi-class classification is the rotten tomatoes movie reviews dataset. This dataset was collected by Pang and Lee [6]. Each and every sentence is parsed into its tree structure and a fine-grained sentiment label ranging from 0-4 is assigned to each node. The labels are interpreted as very negative, negative, neutral, positive, and very positive. There are 5 columns which are: PhraseId, SentenceId, Phrase, and Sentiment. PhraseId starts from 1 until 156060. SentenceId comprises of one number for each sentence and its nodes. Phrase represents the phrase. Sentiment represents the labels of the sentiments from 0-4.

The dataset which we are using is imbalanced. The following Fig. 1 shows the distribution over a bar graph where the label having the highest number of sentences is 2 - 79582, followed by 3 - 32927, followed by 1 - 27273, followed by 4 - 9206, and followed by 0 - 7072. For code snippet for plotting the Fig. 1. is given in Appendix A.

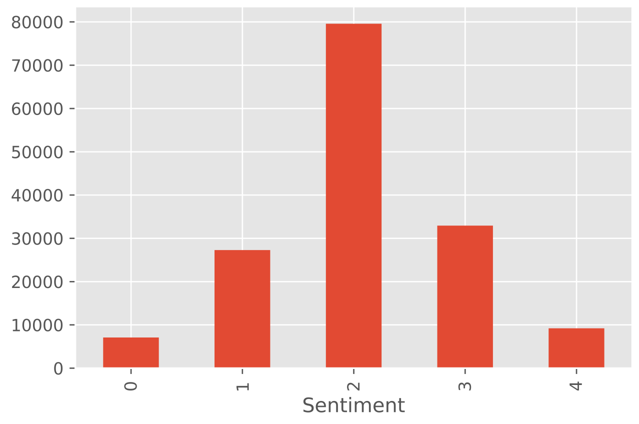

Fig. 1. Imbalanced Dataset.

The Fig. 2. below shows the preprocessing steps that are performed on the above data

IV. Implementation

Keras is a Neural-Network library used. It is preferred for its modularity and ease of use [7]. Pandas library is being used to extract the data and load the data for analysis [8]. Numpy is used for input and analysis of the array object [9]. We also have to convert the Pandas Dataframes into a NumPy array.

A. Bag of Words (BoW)

Bag of Words is one of the simplest to implement approach to NLP. This method considers sentences as string of words and counts the number of recurrences of the given words. Based on the number of times the words appear in the given sentence or paragraph. It works on the assumption that

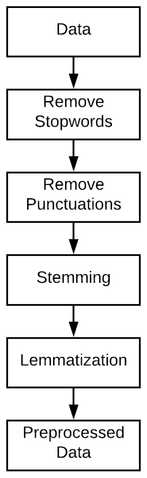

Fig. 2. Preprocessing Steps.

the word that has highest occurrence has higher importance. Decisions for Sentiment are made based by categorizing the high frequency words (also known as term frequency).

B. Term Frequency - Inverse Document Frequency (TF-IDF)

Term Frequency - Inverse Document Frequency is a continuation to above Bag of Words approach. The earlier method used Bag which classified the words in a paragraph by the frequency of occurrence. This method works on the same lines but goes a step ahead by using Inverse Document Frequency, where it checks the words repeated in other documents to eliminate the common words in both documents (words like a, an, the, is, was etc. are eliminated as they have lesser importance than other key words) and thereby, improving the quality of the result.

C. Word2vec

Word2vec is used to produce word embeddings(words or phrases are mapped to real numbers). The input in a Word2vec model is a large corpus of text and the output is a large vector space, typically can be several hundred dimensions, where each unique word is assigned a corresponding vector in space. Word vectors are placed in vector space in such a way that words that share common contexts in the corpus are located located close to one another in the space [10].

V. Proposed Model

The model which I have used while doing this analysis has many different layers. All the layers are imported from the the keras package keras.layers.

- Conv1D
- AvgPooling1D
- Flatten Layer
- Dense Layer

Fig. 3. below shows the proposed model summary.

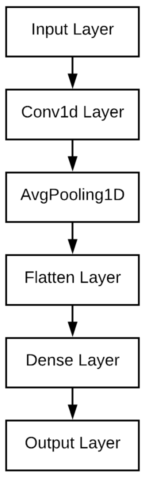

Fig. 3. Model Summary.

For defining the model, the code snippet in Appendix B is used, where I first initialize the variable model to sequential. Once this is done, then use model.add to add different layers to our model.

For compiling the model by giving the loss function and optimizer, the code snippet in Appendix C is used.

VI. Experimental Analysis

While working with the code, several different combinations of methods were applied to it. These methods mean changing different parameters in the code, which include: optimizer, vectorizer, batch_size, learning rate, and more. I experimented by changing the parameters, like, optimizer, vectorizer batch_size, learning rate, and more. The most impact my model had was when I changed the optimizer. In all of the experiments performed, epochs were set to 20.

Following are the metrics [11] which are being used for the analysis:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$F1Score = \frac{2 \times (Recall \times Precision)}{Recall + Precision} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

where TP - True Positive, TN - True Negative, FP - False Positive, and FN - False Negative.

Table I below shows the different results (accuracy, f1 score, precison, and recall) obtained when the vectorizer used is count vectorizer, and the max_features was set to 3000.

Table II below shows the different results (accuracy, f1 score, precison, and recall) obtained when the vectorizer used is tf-idf, and the max_features was set to 3000.

TABLE I
Experimental analysis using different optimizers when vectorizer used is Count Vectorizer.

Optimizer	Results
Adam	Acc: 0.61 F1: 0.58 Precision: 0.65 Recall: 0.53
Nadam	Acc: 0.61 F1: 0.58 Precision: 0.66 Recall: 0.53
Adamax	Acc: 0.61 F1: 0.58 Precision: 0.66 Recall: 0.52

TABLE II
Experimental analysis using different optimizers when vectorizer used is TF-IDF.

Optimizer	Results
Adam	Acc: 0.61 F1: 0.58 Precision: 0.67 Recall: 0.50
Nadam	Acc: 0.61 F1: 0.58 Precision: 0.67 Recall: 0.51
Adamax	Acc: 0.60 F1: 0.56 Precision: 0.69 Recall: 0.49

A. Model Performance

According to the above Table I and Table II, the best performing model is the model that has the following hyper-parameters:

TABLE III
Parameters and their values used.

Parameter	Value
Vectorizer	TF-IDF
Optimizer	Nadam
Loss Function	Categorical Crossentropy
Batch Size	512
Epochs	20
Activation Function	ReLu (For Conv1d layer) Softmax (For Dense layer)

This model has the following results:

- Accuracy: 0.61
- F1 Score: 0.58
- Precision: 0.67
- Recall: 0.51

Fig. 3. Epoch v/s Accuracy. and Fig. 4. Epoch v/s Loss. shows the training history visualization of my best performing model as explained in the keras documentation [12].

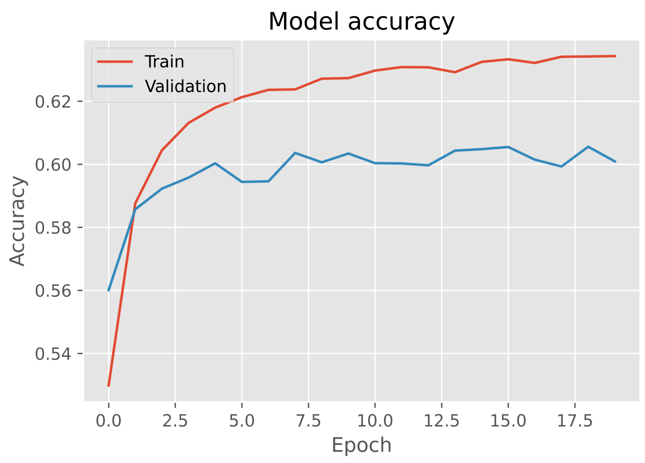

Fig. 4. Epoch v/s Accuracy.

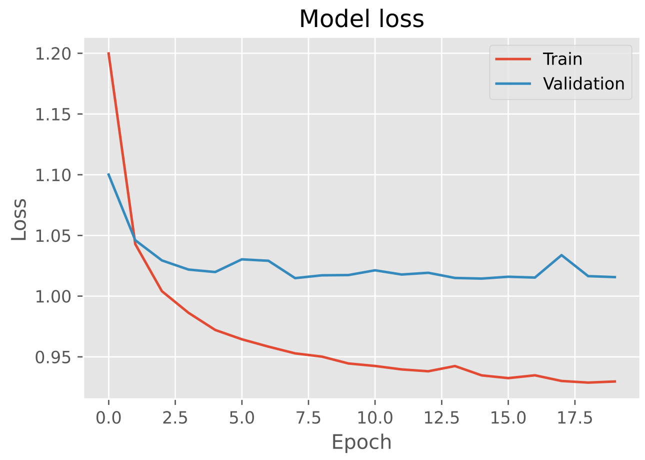

Fig. 5. Epoch v/s Loss.

VII. Conclusion

Multi-class sentiment analysis has been performed on the Rotten Tomatoes movie reviews dataset. The software environment used is Google Colab, and the language used is Python. Before performing any analysis on this data, it is first preprocessed by removing the stopwords, removing the punctuations, stemming, and lemmatization, now the preprocessed data is then used to do further analysis. The data is then split into train data (70%) and test data (30%). Pandas library helps in performing all the above steps efficiently as it converts the data into a dataframe, which is easier to work within Google Colab. Keras (a neural network library) is used to create our CNN model. The model produced is a sequential model. For training purposes, the preprocessed data is passed through that model for a total of 20 epochs. Once all the iterations have been completed, the test data would be used for testing purposes. After a thorough experimental analysis, the best model had the following performance metrics: accuracy: 0.61, f1 score: 0.58, precision: 0.67, and recall: 0.51.

Appendix

A. *Plotting the values of sentiment*

```
# For the bar graph
import matplotlib.pyplot as plt
plt.style.use("ggplot")

fig = df.groupby('Sentiment').Phrase.count().plot.
    bar(ylim=0)

plt.savefig('/content/drive/My Drive/NLP Assignment
    1/assgn2.png', format='png', dpi=1200)
```

Listing 1. Plotting the values of the sentiment.

B. *Model Definition*

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3,
    activation='relu', input_shape=(x_train_np.shape
    [1],x_train_np.shape[2])))
model.add(AveragePooling1D(pool_size=2))
model.add(Flatten())
model.add(layers.Dense(5, activation='softmax'))
```

Listing 2. Model Definition.

C. *Compiling the model*

```
# Used for compiling the model
model.compile(optimizer='nadam', loss='
    categorical_crossentropy', metrics=['accuracy',
    f1_m,precision_m,recall_m])
model.summary()
```

Listing 3. Compiling the model.

References

[1] ”Deep Learning”. wikipedia.com. https://en.wikipedia.org/wiki/Deep_learning (Accessed March 18, 2020).

[2] ”Sentiment Analysis”. monkeylearn.com. https://monkeylearn.com/sentiment-analysis/ (Accessed March 18, 2020).

[3] A. Farooq, S. Anwar, M. Awais and S. Rehman, ”A deep CNN based multi-class classification of Alzheimer’s disease using MRI,” 2017 IEEE International Conference on Imaging Systems and Techniques (IST), Beijing, 2017, pp. 1-6. doi: 10.1109/IST.2017.8261460

[4] A. Rane and A. Kumar, ”Sentiment Classification System of Twitter Data for US Airline Service Analysis,” 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp. 769-773. doi: 10.1109/COMPSAC.2018.00114

[5] B. Chen, Q. Huang, Y. Chen, L. Cheng and R. Chen, ”Deep Neural Networks for Multi-class Sentiment Classification,” 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Exeter, United Kingdom, 2018, pp. 854-859. doi: 10.1109/HPCC/SmartCity/DSS.2018.00142

[6] Bo Pang, Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, p.115-124, June 25-30, 2005, Ann Arbor, Michigan.

[7] “Keras backends”. keras.io. https://keras.io/backend/ (Accessed March 18, 2020).

[8] “Python data analysis library”. pydata.org. https://pandas.pydata.org/ (Accessed March 18, 2020).

[9] “Numpy reference”. scipy.org. https://docs.scipy.org/doc/numpy/reference/ (Accessed March 18, 2020).

[10] ”Word2vec”. wikipedia.com. https://en.wikipedia.org/wiki/Word2vec. (Accessed March 19, 2020).

[11] ”Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures”. exsilio.com. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ (Accessed March 21, 2020).

[12] ”Training history visualization”. keras.io. https://keras.io/visualization/ (Accessed March 21, 2020).