NLP (COMP5413) Assignment 2:Multi-Class Sentiment Analysis With Deep Learning

Khushal Paresh Thaker
Dept. of Computer Scinece
Lakehead University
Thunder Bay, Canada
kparesh@lakeheadu.ca
1106937

Abstract—***The expansion of web has increased the sentiment content present in the web. Understanding the human sentiments towards various entities allows for better recommendation systems and a proper analysis of data available. This paper proposes a multi class CNN (Convolutional Neural Network) model to analyze sentiments of human reviews from Rotten Tomatoes dataset.***

Index Terms-- CNN, Natural Language Processing, Sentiment Analysis, Embedding Layer

I. Introduction

The advent of technology has led to machines learning new ways to perform tasks that human can perform, but quicker. Recently, the domain of machine learning has gained new insights in training model and predicting values which is quite difficult and error prone when performed manually. Sentiment analysis is an important aspect of data mining where in novel data can be analyzed and categorized into various sentiments. The sentiments which are embedded in comments or reviews can be measured and categorized by its polarity [1]. In the modern day, with general public and the critics using the web as a medium to post their reviews, vast data is available online that can be used to analyze the sentiments.

The deep learning model used is CNN which contains shifting convolutional and pooling layers with a convolution filter to perform feature extraction [6]. CNN model typically has small filters on input data. The pooling layer reduces the computational cost of the learning process and also reduces overfitting [7]. CNN takes in the concept of parameterization, where one set of parameters is only used for it to learn instead of having to learn at all the locations [8]. Efficiency increases as there are fewer parameters meaning less computational time. The final layer will be the multi layer perceptron which basically converts the input (extracted features) into the output. The advantage of CNN is that it can be combined into various deep learning architectures where the input of another convolutional layer is the output of the current CNN.

The Rotten Tomatoes dataset used for this model contains more than 156060 phrases, which were parsed from around 8529 complete sentences. There are 5 sentiment labels considered: 0 - negative, 1 - somewhat negative, 2 - neutral, 3 - somewhat positive and 4 - positive.

Having a machine learn how to understand the context like how humans do is difficult. Efficient deep learning algorithms and advancements in natural language processing techniques made it possible to analyze reviews and correlate user’s sentiment towards them.

II. Literature review

The traditional method to perform sentiment analysis was Bag of Words (BOW). Pang et al. [2] proposed a model that used Naïve Bayes, Maximum Entropy and SVM (Support Vector Mechanism) to perform sentiment classification on movie dataset. IMDB Dataset was used resulting with highest accuracy from using SVM mechanism.

Ari et al. [3] proposed a method that combined both the expert review from Rotten Tomato dataset and expert original score using SentiWordnet to extract sentiment score from the dataset.

Liu et al. [4] proposes a model to predict sentiments from a given review using the IMDB dataset. Proposed model has a non-linear regression model for sentiment prediction based on three factors which are reviewer’s expertise, the writing style and timeliness.

Dholpuria et al. [5] compares deep learning model including Convolutional Neural Network with other supervised Machine Learning classifiers like SVM, KNN (K-Nearest Neighbor) and ensemble methods. The proposed CNN model is compared with other models like SVM, KNN and gets the highest F1 Score with 99.345.

Model that involves CNN and LSTM (Long Short-Term Memory) was proposed by Sorostinean et al. [9]. It uses Rotten tomatoes dataset and compares the performance of classification between Naïve Bayes, SVM and CNN and LSTM. According to the model proposed, Naïve Bayes performed better than other models as number of epochs to train the model using CNN and LSTM method took more time and couldn’t be achieved.

III. Proposed Model

The proposed model is a multi-layer, 1D convolution network to classify the sentiments of user comments on the

Rotten Tomatoes dataset using Keras executed in Google Colab. The dataset contains 156060 records with PhraseID, SentenceID, Phrase, and Sentiment as its features as shown in the figure 3. There are many lines of a main sentence which are labelled with 'Sentiment' as well. There are a total of 8529 complete sentences in the dataset. Figure 1 represents a bar plot which depicts that the 2nd Sentiment has the most number of phrases associated with it.

When the complete sentences are taken into consideration, it changes the plot significantly with sentiment 3 and 1 being close to each other and others having a relatively lower count as shown in the figure 2. The dataset is then pre-processed so as to increase the accuracy. The dataset is lemmatized using 'WordNetLemmatizer' and then stemming is performed 'using SnowballStemmer' which are instances of 'nltk.stem'. NLTK (Natural Language tool Kit) provides a tokenizer 'punkt' which is used to divide a text into list of sentences. Random over sampling is performed on the dataset as it is highly imbalanced.

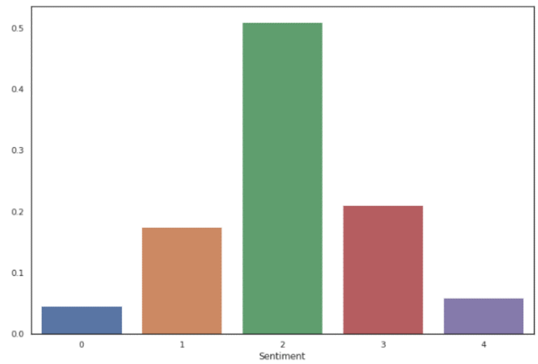


Fig 1: Partition of classes in the dataset

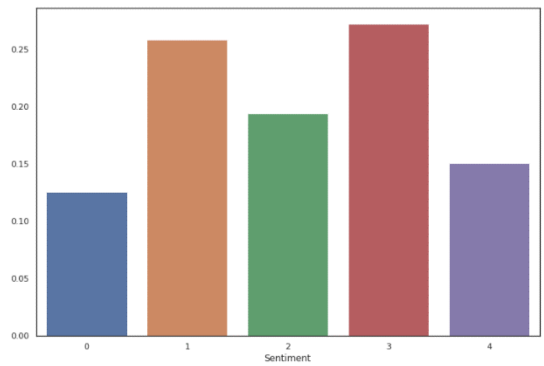


Fig 2: Partition of classes in the dataset with complete sentences

The class of a sentiment is interpreted by exploring it through the words that are composing it. Figure 5 represents a graphical representation of the word frequencies. Word cloud library of python is used for representing the word cloud structure. The dataset is then split into train and test set with a ratio of 70:30 using 'sklearn.model_selection' library with random states set to 2003.

There are stop words, punctuations which are not necessary and have been processed and removed as they have no actual significance towards the overall meaning of the text. Words such as 'the', 'of' occurred most with 35502 and 25748 times in the entire dataset. And multiple words such as 'Yourself', 'blustery', 'chortles' occurred once. The most occurring punctuation was ',' with 34073 occurrences. Most common trigram was ('one', 'of', 'the') occurring 1554 times.

TF-IDF [11] (Term Frequency – Inverse Document Frequency) is a text mining technique which allows categorizing documents by emphasizing words that occur frequently in a document and also discards the importance of words which are present frequently in multiple documents. Also, sklearn's built in vectorisers are used to convert the data into vectors. The tri grams used are within the vectorizer.

Next, the vectorization is performed and data is brought in its vector form. 'to_categorical' is used to convert the dataset into a matrix with as many columns as there are classes. F1 Score, Precision, and Recall are the metrics that are defined which will be used to measure the performance of the model. F1 score is the weighted average of Precision and Recall.

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

Fig 3: Sample Rotten Tomatoes dataset fetched from the provided URL

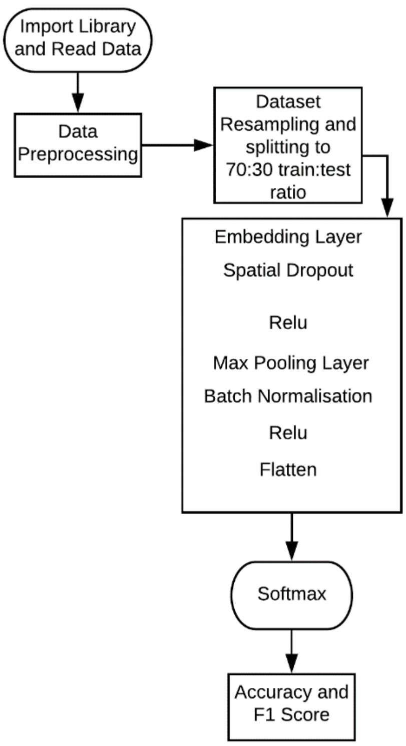


Fig 4: Model Architecture

The Keras model follows the sequential API (Application Programming Interface) allowing a layer by layer creation of model. As shown in the figure 4, the model contains an Embedding layer so that positive integers can be converted into dense vector of fixed sizes. Spatial Dropout layer is added which helps in increasing the independence between feature maps and also prevents overfitting in the data.

Number of filters which is referred to as the kernel size is incorporated along with the max pooling layer. Kernel size is defined as 3 for this model. Max pooling layer is used to reduce the total number of layers. There also exists an average pooling layer, which isn't the best choice for this model. Flatten layer which is used to convert the data into a 1D array is used so that it can be passed as an input to the next layer.

An activation function, ReLU (Rectified Linear Unit) is used to output the input directly if positive, or output zero, if negative [10]. Softmax function is considered as the activation function for the model as it is a multi-class classification problem and softmax's range is between 0 and 1. Before creating the model, the text is converted into sequence of tokens and these sequences are padded so as to have the same length.

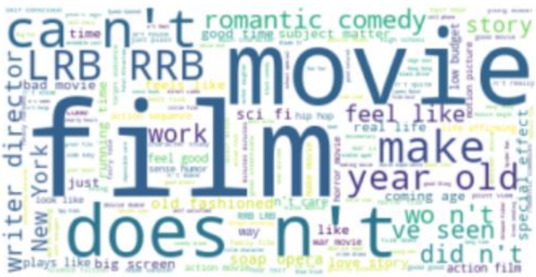


Fig 5: Frequency of words in a positive class

Batch size impacts the efficiency of training and also the noise level of the gradient estimate. For this model, the batch size is maintained at 128 after comparing with that of 64 size. The proposed model is saved as 1106937_1dconv_reg.h5 using 'model.save' property.

The main part of the model lies in the architecture of CNN where it has two layers which are run through the ReLU layer. ReLU is preferred over any other functions such as sigmoid or tanh because these functions have a tendency to saturate whereby the largest value tends to 1 and smallest value tends to -1 for tanh and 0 for sigmoid. This drawback is called as the vanishing gradient problem. ReLU, being a non-linear activation function saturates only uni-directionally and hence, the vanishing gradient problem reduces drastically.

To have the model trained, the epochs are defined to 5. For the defined epoch, the model goes through the batches and gets the average loss, its F1 Score, Precision, Recall and Accuracy. F1 Score is used as a measure because the false negative and the positive values are very important. It is also a better measure that Accuracy because the dataset is highly imbalanced. Figure 6 represents the best model used to implement the problem of sentiment analysis after many iterations. There are a total of 2096743 trainable parameters and 128 non trainable parameters used in the model.

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_8 (Embedding)      (None, 48, 150)           2059650
_________________________________________________________________
spatial_dropout1d_8 (Spatial (None, 48, 150)           0
_________________________________________________________________
conv1d_15 (Conv1D)           (None, 48, 64)            28864
_________________________________________________________________
max_pooling1d_15 (MaxPooling (None, 24, 64)            0
_________________________________________________________________
batch_normalization_8 (Batch (None, 24, 64)            256
_________________________________________________________________
conv1d_16 (Conv1D)           (None, 24, 32)            6176
_________________________________________________________________
max_pooling1d_16 (MaxPooling (None, 12, 32)            0
_________________________________________________________________
flatten_8 (Flatten)          (None, 384)               0
_________________________________________________________________
dense_8 (Dense)              (None, 5)                 1925
=================================================================
Total params: 2,096,871
Trainable params: 2,096,743
Non-trainable params: 128
```

Fig 6: Summary of Simple CNN Model

IV. EXPERIMENTAL ANALYSIS

The given model was split 70:30 for training and testing respectively and is compared with itself when run with two different batch sizes, 64 and 128 respectively. Model is run through 5 epochs and tends to perform better with 128 as its batch size generating 0.8312 as its average accuracy, where as it was 0.7165 with 64 batch size.

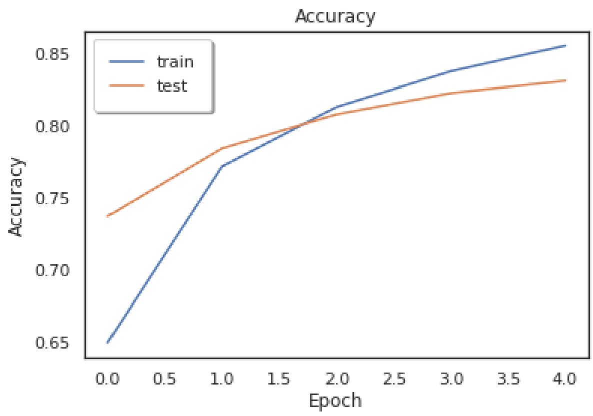
Table 1: Accuracy and Loss for different models implemented

Model	Accuracy	F1Score	Precision	Recall	Loss
1	0.7789	0.736	0.727	0.786	0.587
2	0.7829	0.759	0.746	0.773	0.494
3	0.7863	0.767	0.767	0.767	0.433
4	0.7905	0.768	0.758	0.778	0.412
5	0.7991	0.786	0.766	0.796	0.404
6	0.8103	0.792	0.797	0.791	0.399
7	0.8318	0.8125	0.819	0.8304	0.385

The model was run for 5 epochs with multiple layers being added and removed in order to improve the accuracy. Table 1 represents the Accuracy, F1 score, Precision, Recall and Loss of the models run. Graph 1 depicts the Accuracy between train and the test run. The best model had a gradual increase of validation accuracy from 0.7376 to 0.8318 through 5 epochs. The training accuracy increased from 0.6498 to 0.8561.

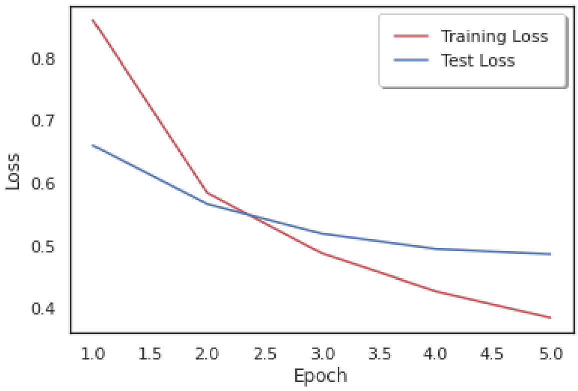
The precision and Recall values were calculated for each and found to be the best for the final model. Model 1 was run with only one convolutional layer and gained a F1 Score of

0.736. With many different iterations and adding a batch normalization layer, the final model showed a F1 Score of 0.8125. Multiple iterations displayed gradual improvement in the accuracy and F1 Score.



Graph 1: Accuracy for Train and Test set

Graph 2 represents the Loss for the model during the train and test run. The best model had a loss of 0.8610 in the 1st epoch and gradually decreased to 0.3856 with the 5th run. Other models had a average loss of 0.492 from 5 epochs.



Graph 2: Loss for Train and Test set

CONCLUSION

The proposed model is a 1D Convolution based neural network to analyse the sentiment of movie reviews from the Rotten Tomatoes dataset. The model was tested with 30% dataset. The performance measure used is the F1 Score along with Precision and Recall. The best accuracy of the final model was 0.8318 with an average Loss of 0.392 run through 5 epochs with 128 as the batch size, comparatively better than the model when run with 64 as the batch size or even with different combinations of convolutional layers. The code is run in google colab using python. The rotten tomatoes dataset was run with a Bi-LSTM model and a CNN model by Mohamed et al [9] which provided accuracy less than 50% as they didn't perform the necessary preprocessing and also needed more computational power and lot of time to run the code.

REFERENCES

[1] *Dey, Lopamudra, Sanjay Chakraborty, Anuraag Biswas, Beepa Bose, and Sweta Tiwari. "Sentiment analysis of review datasets using naive bayes and k-nn classifier." arXiv preprint arXiv:1610.09982 (2016).*

[2] *Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pp. 79-86. Association for Computational Linguistics, 2002.*

[3] *Firmanto, Ari, and Riyanarto Sarno. "Prediction of movie sentiment based on reviews and score on rotten tomatoes using SentiWordnet." In 2018 International Seminar on Application for Technology of Information and Communication, pp. 202-206. IEEE, 2018.*

[4] *Liu, Yang, Xiangji Huang, Aijun An, and Xiaohui Yu. "Modeling and predicting the helpfulness of online reviews." In 2008 Eighth IEEE international conference on data mining, pp. 443-452. IEEE, 2008.*

[5] *Dholpuria, Tanushree, Y. K. Rana, and Chetan Agrawal. "A Sentiment analysis approach through deep learning for a movie review." In 2018 8th International Conference on Communication Systems and Network Technologies (CSNT), pp. 173-181. IEEE, 2018.*

[6] *Le Zhang P.N Suganthan, A Survey of randomized algorithms for traininig neural networks, Information Sciences, Volumes 364-365, 2016*

[7] *L Sayavong, Z. Wu and S. Chalita, "Research on Stock Price Prediction Method Based on Convolutional Neural Network," 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), Jishou, China, 2019, pp. 173-176.*

[8] *Liu, Weibo, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. "A survey of deep neural network architectures and their applications." Neurocomputing 234 (2017): 11-26.*

[9] *Sorostinean, Mihaela, Katia Sana, Mohamed Mohamed, and Amal Targhi. "Sentiment analysis on movie reviews." In Journal Agroparistech. 2017.*

[10] *Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." arXiv preprint arXiv:1803.08375 (2018).*

[11] *Ramos, Juan. "Using tf-idf to determine word relevance in document queries." In Proceedings of the first instructional conference on machine learning, vol. 242, pp. 133-142. 2003.*