NLP Assignment 2: Multi-class Sentiment Analysis using Deep Learning

Palak Bhatia
Lakehead University)
Email-pbhatia3@lakeheadu.ca
Student Id-1116552

***Abstract*—Sentiment analysis is acknowledged as one of the most significant sub-areas of Natural Language Processing (NLP) research where comprehension of implied or explicit emotions conveyed online is beneficial to consumers, company owners, and other stakeholders. It provide the simple consumer view, which may be something that carries a contextual conclusion, such as an internet analysis, reviews on blog entries, film ranking, and so on. This paper targets 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 used is from the Rotten Tomatoes movie review corpus that has been greatly improved upon,and annotated with a fine sentiment score via Amazon Mechanical Turk.**

I. Introduction

Nowadays it has become very fashionable to share opinions and post comments regarding locations seen or films shown. It has contributed to the desire to make use of the large volume of data immediately. Therefore, the human language is complex to teach a computer to interpret the specific grammatical complexities, cultural differences, slang and incorrect spellings that exist in user feedback is a difficult operation. It is much more difficult to train a computer to consider the importance of a tone of an entity can be. Advances in machine learning and the analysis of natural language techniques have allowed user feedback to be evaluated and user preferences to be recognised. Such techniques of evaluating emotions are effective in a number of fields, such as industry or politics.[4] The aim of this assignment is to use the Rotten tomatoes movie review corpus to experiment with different approaches of deep learning for the task of sentiment analysis. The data set is picked from the website called Rotten Tomatoes and it has four columns which are phrase Id, phrase, sentence Id and sentiment. More precisely, data set phrases must be divided into 5 groups according to the user's desired tone: negative, somewhat negative, neutral, somewhat optimistic, optimistic[1]. Here I have implemented a deep learning model using keras that uses convolution, activation and max-pooling layers for analysing the dataset. The dataset is initially pre-processed using various techniques such as tokenization, lemmatization, removal of stop words, normalization etc. Then the text format of the data is converted into numerical format that is easily understood by the machine, hence we use TF-IDF (term frequency-inverse document frequency) vectorization. The pre-processed dataset is finally trained using the build model to perform multi-class sentimental analysis.

Identify applicable funding agency here. If none, delete this.

II. Literature Review

The sentiment can be described as a perception of a circumstance or case, or an attitude towards it. It also entails many kinds of emotions of self-indulgence: joy, tenderness, sorrow, or nostalgia. The sentiment labels may be characterised as polarity or valence (e.g., positive, favourable, and negative) or multiple forms of emotional feelings (e.g., furious, joyful, sad, or proud)[1].The deep learning method, one of the strategies of machine-learning, has recently been commonly used to identify emotions. Primarily, there are two dominant types of deep learning technique: RNN and CNN for sentiment classification. Throughout this assignment, we are introducing a CNN model, whose framework was specifically crafted for successful analysis of multi-class sentiments. Among the current research that use deep learning to identify documents, the CNN takes advantage of the so-called convolutionary filters that automatically learn correct features for the task at hand. A Convolutionary Neural Network (ConvNet / CNN) is a Deep Learning algorithm capable of capturing an input image, assigning significance (learnable weights and biases) to different aspects / objects in the image, and being able to distinguish one from another. The pre-processing needed in a ConvNet is significantly lower than in other classification algorithms.[1]

The main step in training a model involves feeding the model with a clean and pre-processed dataset. Preprocessing of a text dataset involves some steps called as tokenization, removal of stop words, stemming or lemmatization and POS tagging. Once the pre-processed text data is obtained,it needs to be broken down into a numerical format where the computer can interpret easily. BoW(Bag of Words) ,Word2Vec and TF-IDF are techniques which enable us to translate text sentences to numerical vectors. Here in this assignment,we have used TF-IDF (term frequency-inverse document frequency) which is a statistical measure which assesses how important a word is to a subject in a subject set[2]. That is achieved by combining two metrics: how many times a term occurs in a text, and the word's reciprocal frequency in text across a series in documents. It proposes the word's definition to the record and to the whole corpus. Term recurrence educates about the

recurrence of a word in a database, and IDF illuminates the recurrence of the same word in the corpus. The basic flow diagram of a fundamental NLP application is shown in fig1.

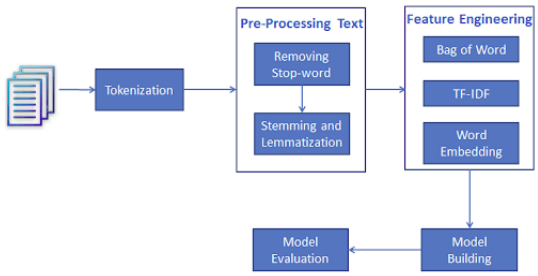

Fig. 1. Flow Diagram for Sentiment Classification

III. Proposed Model

The proposed model makes use of different layers of convolution network combined with an optimizer and activation function to perform multi-class sentiment analysis on the provided dataset. The data provided to the model has been cleaned, pre-processed using tokenization, stemming etc and vectorized using TF-IDF. The model has been trained and tested using the python's keras library.

The algorithm begins by loading the dataset and splitting the dataset in the ratio of 70:30 of training and testing using the sklearn.model selection library with random state is set to 2003. The model consists of two convolution layers followed by max pooling layers. After that, we have a dropout layer(with rate = 0.5) followed by a flattening layer and finally a dense layer which uses softmax activation function. The first convolution layer uses 64 filters having kernel size of 3x3, stride rate of 1 and zero padding. The second convolution layer has 32 filters with kernel size of 5x5. The model summary is shown below in fig2.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_9 (Conv1D)            (None, 1998, 64)          256
_________________________________________________________________
max_pooling1d_9 (MaxPooling1 (None, 1998, 64)          0
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 1994, 32)          10272
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 1994, 32)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 1994, 32)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 63808)             0
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 319045
=================================================================
Total params: 329,573
Trainable params: 329,573
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. Model Summary

The flattening layer converts it into a column vector. The use of the ReLu layer was to convert all the negative values to zero. By using the Conv1d layers with the ReLU activation function, the model becomes non-linear. I have used max-pooling, to pick the maximum value as a representation of the peripheral values. As the emotion is always defined by a mixture of many terms rather than voicing the meaning in each word in the paragraph, I have followed the methodology of max-pooling. Because max-pooling is the layer that transfers the maximum value between many values to the next layer, this results in performance vectors of a much smaller size. I have used the softmax activation layer in the final dense layer so as to avoid binary classification and to accomodate 5 classes in my model. The softmax function outputs, the value, which is the probability value, is generated for each class.The following subsections discuss the various elements of the model.
The proposed model has been trained over 100 epochs and for each set the batch size have been set to 64. The optimizer used in the model is Adam optimizer having learning rate as 0.001. The final hyper-parameters obtained after tuning are shown in below Table I.

TABLE I
Optimized Hyper-parameters after tuning

Hyper-parameter	Value
Vectorization Method	TF-IDF(with Unigram)%
Optimizer	Adam%
Batch Size	64%
Learning rate	0.001%
Number of Epochs	100
Final activation function	Softmax
Loss Function	Categorical Cross Entropy

A. Dataset

The data provided for this challenge is comprised of phrases extracted from the Rotten tomatoes dataset. The training data contains more than 156000 phrases, which were parsed from about 8500 sentences by the Stanford parser. Each phrase has an associated sentiment label[3]. There are 5 sentiment labels considered:

- 0 - negative
- 1- somewhat negative
- 2-neutral
- 3- somewhat positive
- 4-positive

The dataset contain 4 attributes which are shown in Table II. The first 10 rows of the dataset are show below in fig3.

TABLE II
Attribute Description

Attribute Name	Data Type	Description
PhraseId	Int	Unique Id for each phrase
SentenceId	Int	Unique Id for each sentence
Phrase	Char	Movie review phrases
Sentiment	Int	Sentiment labels for the phrase

B. Libraries

I have used the pandas library to load the dataset and perform initial pre-processing of the dataset. For converting the

	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
5	6	1	of escapades demonstrating the adage that what...	2
6	7	1	of	2
7	8	1	escapades demonstrating the adage that what is...	2
8	9	1	escapades	2
9	10	1	demonstrating the adage that what is good for ...	2

Fig. 3. First 10 rows of the dataset

dataset into array and performing mathematical calculation, I have used numpy. Also, to make use of various functions such as tokenization, lemmatization, removing stop words, I have utilised the NLTK library. The sklearn library has been used to perform the split over dataset into 70:30 with a random state set to 2003. Finally I have make use of keras library to build and train my model.

C. Preprocessing

After analysing the data, I found that the stop words, punctuation, unicode, and uppercase characters are present. Initially, I converted the document into tokens using tokenization and then begun removing stop terms and punctuations as they do not have any context for the general sense of the text. I have removed the punctuation and transformed unicode to ASCII characters and finally I have lemmatise the tokens in the corpus. For the input layer, I feed in the word vector representations for unigrams.

D. Vectorization

I have performed TF-IDF vectorization in the solution. TF-IDF(Term Frequency-Inverse Document Frequency) is a text mining technique used to categorize documents. It is a method for emphasizing words that occur frequently in a given document, while at the same time de-emphasising words that occur frequently in many documents. It educates about the recurrence of a word in a record and illuminates about the recurrence of the specific word in the entire corpus[4].

E. One-hot label encoding

I have used hot label encoding to pre proccess the multi class variables. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.This is shown in fig4.

```
Y_train = keras.utils.to_categorical(Y_train, num_classes)
Y_test = keras.utils.to_categorical(Y_test, num_classes)
print(Y_train)

[[[1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0.]
  [1. 0. 0. 0. 0.]]

 [[1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0.]
  [1. 0. 0. 0. 0.]]

 [[1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]]

 ...
```

Fig. 4. One-hot label encoding

F. Over/under fitting issue

Overfitting occurs when the variability of the data is identified by a mathematical model or machine learning algorithm [4]. Since,the max-pooling layer pools out only the maximum data from the kernel, thus reducing the data and eventually minimising the overfitting. To avoid underfitting, I have added two layers of convolution and two layers of max-pooling, an activation layer and linear layers.

G. Number of trainable parameters.

Various learning parameters have been used tuned in the model which are discussed below:

- Kernel: - In machine learning, a “kernel” is usually used to refer to the kernel trick, a method of using a linear classifier to solve a non-linear problem. The model uses a kernel size of 3 and 5 in two layers.
- Batch Size:- A smaller mini-batch size (not too small) usually leads not only to a smaller number of iterations of a training algorithm, than a large batch size, but also to a higher accuracy overall, i.e, a neural network that performs better, in the same amount of training time, or less.Therefore, the batch size used in the model is 64.

- Number of Epochs:- The number of epochs is the number of complete passes through the training dataset. It is set to 100 for the model since the accuracy remains constant even if we keep on increasing the epochs.
- Stride Rate:-In CNNs, a stride represents the number of shifts of pixels in the input matrix. Here the stride rate is set to 1
- Learning rate:-The amount that the weights are updated during training is referred to as the step size or the "learning rate." Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0 . For this model, the learning rate is set to 0.001.
- Optimizer:- Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. Adam optimizer have been used in the model.

IV. Experimental Analysis

The model uses given below metrics to evaluate the model:-

- Accuracy- It is calculated as follows: Accuracy = Correct Predictions / All Predictions It is a measure of percentage value for correct predictions of data.
- Precision- It is a measure for which model is correctly predicting any particular category. It is calculated by following formula: Precision = particular category predicted correctly /all category predictions.
- Recall- Recall is calculated by following formula: Recall = Correctly Predicted Category / All Real Categories
- F1_Score(Figure-of-merit)-It is a harmonic mean of precision and recall.

The Table III below shows the evaluation of the model proposed.

TABLE III
PROPOSED MODEL METRICS

Metric	Value
Accuracy	61.12%
Precision	66.079%
Recall	51.98%
F1_Score	58.05%

While training the model, I altered various training parameters such as kernel size, padding, stride rate, batch size, learning rate, number of epochs etc to achieve higher accuracy. Along with the training parameters, I also changed the optimizers and activation functions to predict the model withn highest testing accuracy. I also tried to compare the accuracy by adding and removing various layers. Also I have changed various vectorization functions and pre-processing steps to achieve higher accuracy. Some of these comparisons have been analyzed below:-

- Vectorization function- I have used Continuous Bag-of-Words Model, Word2Vec and TF-IDF for text vectorization of the data and achieved highest accuracy using TF-IDF as shown in below Table IV.

TABLE IV
ACCURACY USING VARIOUS VECTORIZATION TECHNIQUES

Vectorization Function	Accuracy
TF-IDF(unigram)	61.12%
Bag-of-Words	50.12%
Word2Vec	51.98%

- Class Imbalance- The training data contains more than 156000 phrases, and Stanford parser evaluated each from around 8500 phrases. Each statement has a sentiment mark associated with it. Figure 5 shows the partition of classes in the dataset[3]. As shown, the dataset depicts high class imbalance and therefore I have used class weights while evaluating the model which increases the accuracy to 61.11%.

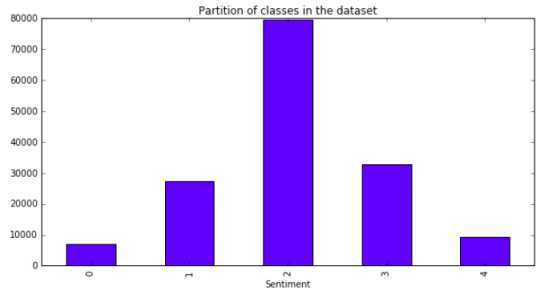

Fig. 5. Partition of classes in the dataset

- Optimizer- Using different optimizers such as Adam, Adamax and SGD gave different accuracies, where Adam being the best. It gave the testing accuracy of 61.12%. SGD optimizer gave just 29.55% accuracy whereas adamax gave 40.63% as shown in Table V below.

:

TABLE V
ACCURACY USING VARIOUS OPTIMIZERS

Optimizers	Accuracy
Adam	61.12%
Adamax	52.12%
SGD	48.08%

- Batch Size:- Accuracies using different batch sizes are shown in Table VI below.

TABLE VI
ACCURACY USING DIFFERENT BATCH SIZES

Batch Size	Accuracy
64	61.12%
32	54.09%
128	51.11%

- Inference Time:- This inference time obtained for the best-trained model which gives accuracy of 61.12% is 3742.7446 sec.

- Model Storing and Loading:- The proposed model is being stored using keras in a json format. Also, the github link for the code can be accessed using https://github.com/palakbh18/NLP_Assignment2. The same model can be loaded and evaluated to give the results as shown in fig6.

Fig. 6. Saving and evaluating model

V. Plots

- Training metric plots:- The below Fig 6,7,8,9,10 depicts the plots between accuracy versus epochs, precision versus epochs,loss versus epochs, recall versus epochs and F1_Score versus epochs respectively. The plots has been created using the matplot library of python.

Conclusion

A non-linear deep learning model has been implemented using convolution neural network for performing multi-class sentiment analysis on the rotten tomatoes movie dataset. Appending a softmax layer at the end of the fully connected provides a solution for doing multi-class analysis. Preprocessing the dataset using tokenization, lemmatization and removal of stop words and punctuations helps in boosting the performance. On the basis of above observations, we see that a CNN model having 2 convolutional layers, max-pooling layers, ReLU activation function and softmax layer with batch size = 64, optimizer = Adam, number of epochs=100 and TF-IDF vectroization provide the highest testing accuracy of 61.11%.

References

[1] Sentiment Classification Using Convolutional Neural Networks by Hannah Kim and Young-Seob Jeong,IEEE 2019
[2] Predicting Sentiment from Rotten Tomatoes Movie Reviews by Jean Y. Wu and Yuanyuan Pao; IEEE 2018
[3] Implementation of n-gram Methodology for Rotten Tomatoes Review Dataset Sentiment Analysis by Prayag Tiwari, Brojo Kishore Kishore Mishra, Sachin Kumar and Vivek Kumar; International Journal of Knowledge Discovery in Bioinformatics, June 2017
[4] Sentiment Analysis on Movie Reviews,Mihaela Sorostinean, Katia Sana, MOHAMED Mohamed, Amal Targhi; IEEE 2017

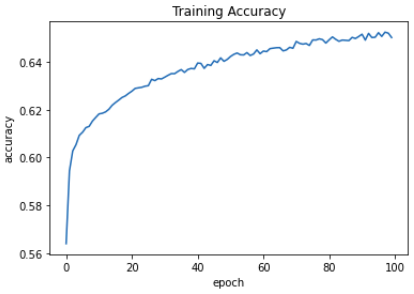

Fig. 7. Training Accuracy versus number of epochs

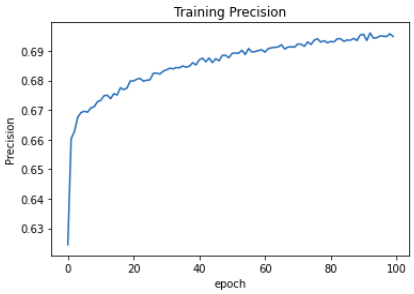

Fig. 8. Training Precision versus number of epochs

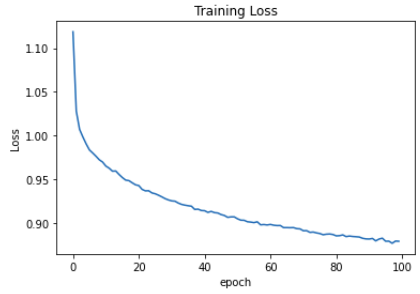

Fig. 9. Training Loss versus number of epochs

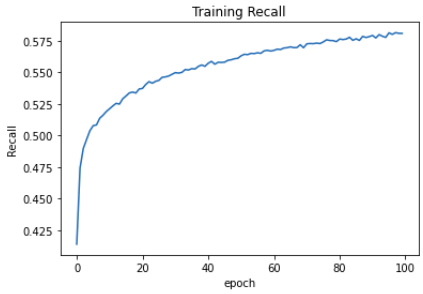

Fig. 10. Training Recall versus number of epochs

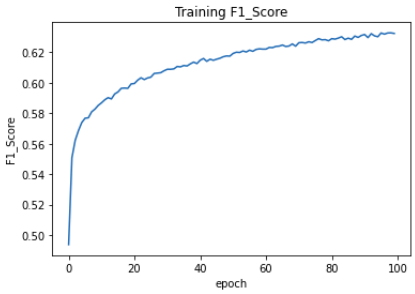

Fig. 11. Training F1_Score versus number of epochs