Multi-Class Sentiment Analysis using Deep Learning

Jagmeet Singh Sidhu
Department of Computer Science
Lakehead University
Student id. 1102275
jsidhu10@lakeheadu.ca

***Abstract*—This paper is targeted to finding the most effective One dimensional convolution neural network for predicting the Sentiment using the Movie review dataset. In order to predict the sentiments value I have used this famous Rotten Tomatoes movie reviews which was available on GitHub. I have build a Multi class classifier model for predicting the attribute which is sentiments. This CNN model is build up of 3 convolution layer and used RELU as the activation function. I have done the hyper-parameter tuning in order to find the value of parameters for which model gives the lowest Loss. After getting best hyper-parameters I am able to achieve the training accuracy of 70% and testing accuracy by 62%.**

***Index Terms*—Convolution Neural Network, Tensorflow, Keras**

I. Introduction

Convolution neural network is not only used for developing computer vision application such as image classification but also it is used solving the problems related to Natural Language Processing(NLP). CNN is nothing just several layers of convolution which uses RELU as an activation function. RELU is a non- linear activation function hence it makes the model build becomes non-linear model.
This paper talk about classifying the sentiment based on the attributes such as moview reviews,Phrase, Sentence etc. In order to predict the value 1D-CNN is implemented.

II. Literature Review

Convolution neural network has a wide application in the domain of machine learning, image classification and natural language processing. CNNs are like NN which are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output[3]. Each Layer takes the input from the previous layer and preserves the relationship between the attributes. After doing number of convolution it passes throught the RELU which performs non-linear operation. Relu is important as it bring the non linearity in our conv-net. The output of RELU becomes the input for the pooling layer. There are three types of pooling:

- Sum Pooling
- Avg Pooling
- Max Pooling

Out of the above types of pooling, majorly max pooling is used. After doing pooling we flatten our matrix vectors feed into the fully connected neural network. Finally, we have an activation function such as softmax to classify the outputs.[5] Similarly, in this assignment I have used the CNN for classifying the sentiments.

III. Proposed CNN Model

A. Feature Techniques

TF-or(Term Frequency(TF)— Inverse Dense Frequency(IDF)) is a method used to evaluate the meaning of sentences composed of terms and to remove the inability of the Bag of Terms method to interpret text or to help a computer read numerical phrases.
Word2vec is a collection of similar models used to produce embedding of terms. Word2vec takes a broad corpus of text as its input and produces a vector space, usually several hundred dimensions, with each single word in the corpus being given a corresponding vector of space.

B. Dataset

Rotten Tomatoes movie review dataset is used to implement CNN. This dataset contain the Phrases, Sentences in the form of moview reviews etc. This dataset contain 4 attributes which are shown in table 1.

TABLE I
Detail about attribute

Attribute Name	Data type	Description
PhraseID	Float64	The categorical value that helps to know from which phrase it has been taken
SentenceID	Float64	Same like phrase id specifically given to categorize the phrases used
Phrase	String	This is basicaaly the moview review in the sentences form; This is the main sentence where we actually implementing text pre-processing
Sentiment	Float64	The final class value that we going to use for categorization

	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. 1. First ten instances of the dataset.

C. Libraries

In order to do the classification using 1D-CNN, I have used the pandas library to load the data and do some pre-processing part. In order to do mathematical calculation or converting the pandas data-frame to array I have used numpy. Finally, in order to develop the CNN model I have used the TensorFlow framework. So, below is the list of library used:

- pandas
- numpy
- random
- sklearn
- matplotlib

D. Coding

I am using Google Colaboratory which has free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser. I have load the data into pandas data frame using URL method. I have used this URL method because by using this I don't have to upload the data set every time to the Colab directory. The snippet of the code is show in figure 2.

After getting all the data into pandas dataframe I have displayed the first 10 rows of the data using the head function. The output of the code is show in figure 1. In order to split the data I have used the sklearn library which helped me to divide the dataset into testing and training along with value of random state to be 2003.

```
# Imported the necessary libraries
import csv
import urllib.request as urllib2

import matplotlib.pyplot as plt

import pandas as pd
# Importing data using url
url = 'https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/master/train.tsv'
response = urllib2.urlopen(url)

# reading data using pandas and converting into dataframe
df = pd.read_csv(response,delimiter='\t',encoding='utf-8')
df.head(10)
```

Fig. 2. Load Dataset using URL

E. Trainable Hyper Parameters

Hyper-parameters are the variables which determine how the network is trained. Hyper-parameters are set before contracting the model. In my model following are the hyper-parameters:

- Learning Rate- 0.002 to 0.0000001
- Epochs- 0 to 12
- Batch Size- 64, 128
- Optimizer- Adamax
- Kernel Size- 3

After analysing the test score for different combination of the hyper-parameters, I have received highest accuracy score when learning rate was 0.002, optimizer was Adamax, epoch was 12 and batch size was 128. I revied training accuracy of 70% and for the testing dataset 62%.

F. Data Pre-Processing

Stemming helps to simplify words to their stems with trailing. A word stem does not need to be the same root as a morphological root based on a dictionary, it is either an equivalent or lesser form of the word. Commonly, stemming

```
for l in range(len(documents)):                    #For each review document
  label = documents[l][1]                          #Save review label
  tmpReview = []                                   #Placeholder list for new review
  for w in documents[l][0]:                        #For each word this is review
    newWord = w                                    #Set newWork to be the updated word
    if remove_stopwords and (w in stopwords_en):   #if the word is a stopword & we want to remove stopwords
      continue                                     #skip the word and don't had it to the normalized review
    if removePuncs and (w in punctuations):        #if the word is a punc. & we want to remove punctuations
      continue                                     #skip the word and don't had it to the normalized review
    if useStemming:
      #if useStemming is set to True
      #Keep one stemmer commented out
      #newWord = porter.stem(newWord) #User porter stemmer
      newWord = lancaster.stem(newWord) #Use Lancaster stemmer
    if useLemma:
      newWord = wordnet_lemmatizer.lemmatize(newWord)
    tmpReview.append(newWord)                      #Add normalized word to the tmp review
  documents[l] = (tmpReview, label)                #Update the reviews list with clean review
  documents[l] = (' '.join(tmpReview), label)

print(documents[0])
```

```
('female', 2)
```

Fig. 3. Removing Stop words using Stemming Technique

algorithms are based on rules. We can describe them through heuristic mechanism that sort of lops off the ends of words. A word is analysed and passed through a sequence of conditionals that decide whether it will be cut down or not. I have used the raw train data of Rotten Tomatoes movie reviews and

loaded into pandas dataframe. The proposed CNN model takes an array as input. So, in order to convert the dataframe into pandas I have used the sklearn library which helps to convert the dataframe into numpy array. The numpy array is still not normalized. As normalization is important as it helps in faster convergence on learning rate and also more uniform influence for all weights. So, in order to perform the normalization, I have used the same sklearn library which helps me to perform the normalization. In order to perform testing and training, I have split the numpy array into 70:30 ration where 70% of the data is used for training and rest 30% is used for testing. Splitting of the data is show in figure 4.

```
#Tokenizing Data after Splitting
X_train, X_test, Y_train, Y_test = train_test_split(df ['Phrase'], df ['Sentiment'], test_size=0.3, random_state=2003)
documents=[]
X_train = np.array(X_train.values.tolist())
Y_train = np.array(Y_train.values.tolist())
for i in range(len(X_train)):
  documents.append([list(word_tokenize(X_train[i])), Y_train[i]])

X_test = np.array(X_test.values.tolist())
Y_test = np.array(Y_test.values.tolist())
for i in range(len(X_test)):
  documents.append([list(word_tokenize(X_test[i])), Y_test[i]])

documents[0]
```

```
[['ca', "n't", 'quite'], 2]
```

Fig. 4. Splitting the data into 70:30 ration

G. Proposed Model

I have created a CNN model with one function: feed. The feed() function includes the pre-processing steps outline above and also compute the a forward pass for the CNN. So, when the instance of the class, CnnRegressor, is created, I define internal functions to represent the layers of the net. During the feed pass, we call these internal functions. I have used 4 major function which are:

- model.add (Conv1D (filters=64, kernel_size=3, activation='relu', input_shape=(2000,1))) – applies convolution
- model.add (Conv1D(128,kernel_size=5,activation='relu')) – applies ReLU
- model.add.MaxPool1d(pool_size = 1) – applies max pooling
- model.add(Flatten())– fully connected layer (Flatten Layers inputs by learned weights)

After the CNN model is ready, we need build training model. In order to make training model, we need to find the best and optimum hyper parameters which gives the best results for our proposed model. After doing Hyper-Parameter tuning, I have came up with the optimum hyper-parameters which are shown in table 2.

TABLE II
OPTIMUM HYPER-PARAMETERS FOR PROPOSED MODEL

Hyper-Parameter	Value
Learning Rate	0.002
Optimizer	Adamax
Epoch	12
Batch Size	128
Number of Layer	3
Testing accuracy	test acc
Training accuracy	acc

From the table 3 it can observed that the adamax optimizer works well and give the highest accuracy.

TABLE III
COMPARISON OF OPTIMIZER AND LEARNING RATE

Learning Rate	Optimizer	Training accuracy	Testing accuracy
0.00001	Nadam	0.68	0.58
0.001	Adamax	0.62	0.52
0.00001	Nadam	0.65	0.59
0.002	Adamax	0.70	0.62

H. Over/under fitting issue

The most common challenge in the domain of machine learning is over-fitting. Over-fitting occur when the proposed model works well on the training dataset but not well on testing dataset. When model is not able to predict the correct value for the new un-seen/ new instance is called over-fitting. In order to, overcome the problem of over fitting, I gradually increased the number of layers until there was no major increase in the accuracy and decrease in Loss.
In order to overcome the issue of under fitting, I have increased the complexity layer of model from one layer to two layer of convolution and also increase the value of hyper-parameter such as number of epochs.

I. Vanishing/exploding gradient issue

In order to overcome the issues of the vanishing gradient problem, I have used the small learning rate. As, we are using GPU it wont be be an big issues to perform computation. Also one can use the Relu activation function as it keeps the network computations close to the identity function.

J. Model Storing

After creating the 1d-CNN model, I have stored the model into the drive. In order to save the model I have first connected my drive with the colab using the code displayed in the figure . Stored the model with .h(header file) extension.

```
model.save('1102275_1dconv_reg.h5') #SAVING THE MODEL
model1 = load_model('1102275_1dconv_reg.h5')# LOADING THE SAVED MODELmodel1.layers
```

Fig. 5. Snippet for storing the code

ACKNOWLEDGMENT

I want to deeply thanks my professor Dr. Thangarajah Akilan who helped me understand the concept of CNN and help me in my class labs. I also want thank the TA's Andrew and Punardeep who took keen interest and solved my issues and concerns about the assignments on time.

Conclusion

In this assignment, I have used the the raw train data of Rotten Tomatoes movie reviews dataset containing the more than twenty thousand instances and 4 attributes and one target attribute. Out of the 4 attribute. I have classified the Sentiments using Convolution Neural Network. After hyper-parameter turning, I am able to achieve the highest testing accuracy of 62%.

References

[1] https://www.tensorflow.org/tutorials/keras/classification
[2] https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/master/train.tsv
[3] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
[4] https://stackoverflow.com/questions/51700351/valueerror-unknown-metric-function-when-using-custom-metric-in-keras
[5] Chenyou Fan,'Survey of Convolutional Neural Network'