Multi-Class Sentimental Analysis Using Deep Learning

Pranav Bhatt
Department of Computer Science
Lakehead University
Student id. 1109651
pbhatt3@lakeheadu.ca

***Abstract*—This paper is targeted is to implement Natural Language processing algorithms to predict the sentiment of the reviews from the improved corpus that has the additional sentiment information of all sub-phrases. We used sequential model as it allows us to create models layer-by-layer for most problems. Also we used convolution 1D as CNN's are basically several layers of convolutions with non-linear activation functions like Relu or softmax. During the training phase, a CNN automatically learns the values of its filters based on the task that we want to perform. We split the data into 70:30 ratio where 70 percent is for training and 30 percent for testing. Using the Keras library we implemented the code and got an accuracy of 60 percent with batch size of 64 and 12 epochs.**

***Index Terms*—Convolution Neural Network, Keras, NLP, Softmax, Relu.**

I. Introduction

Sentiment analysis models detect polarity within a text i.e a positive or negative opinion, Weather it's a whole document, paragraph, sentence or a clause. By automatically analyzing customer feedback, from survey responses to social media conversation, brands are able to listen attentively to their customers and tailor products and services to meet their needs. Applications that use sentiment analysis are social media monitoring, brand monitoring and customer service and market analysis. It is estimated that 80 percent of world data is unstructured, so here the sentimental analysis helps businesses make sense of all unstructured text by automatically tagging it. The benefits include Real time analysis; processing Data at scale at a large scale and consistent criteria is estimated. l.
Comparing with many other fields, advances in Deep Learning have brought Sentiment Analysis into the foreground of cutting-edge algorithms. The work has been processed in few steps: Data collection, Text preprocessing, Word Embedding, Training and testing, Sentiment Detection, Sentiment Classified and Results..

II. Literature Review

We have seen a massive increase in the number of papers focusing on sentiment analysis and opinion mining during the recent years. According to our data, nearly 7,000 papers of this topic have been published and, more interestingly, 99 percent of the papers have appeared after 2004 making sentiment analysis one of the fastest growing research areas. Although the present paper focuses on the research articles of sentiment analysis, we can see that the topic is getting attention in the general public, as well. Figure 1 shows the increase in searches made with a search string "sentiment analysis" in Google search engine.

We observed that the first academic studies measuring public opinions are during and after WWII and their motivation is highly political in nature. The outbreak of modern sentiment analysis happened only in mid 2000's, and it focused on the product reviews available on the Web, e.g. Since then, the use of sentiment analysis has reached numerous other areas such as the prediction of financial markets and reactions to terrorist attacks. Additionally, research overlapping sentiment analysis and natural language processing has addressed many problems that contribute to the applicability of sentiment analysis such as irony detection and multi-lingual support. Furthermore, with respect to emotions2 , efforts are advancing from simple polarity detection to more complex nuances of emotions and differentiating negative emotions such as anger and grief.

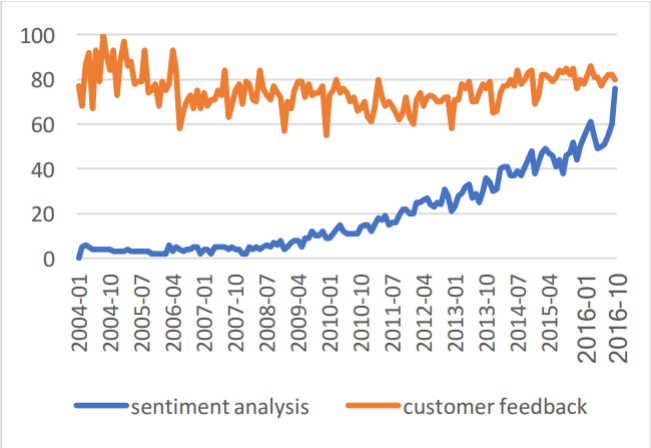

Fig. 1. data showing the relative popularity of search strings "sentiment analysis" and "customer feedback

A. Dataset

The dataset contains the reviews of movies on Rotten Tomatoes which include columns like Phraseid, SentenceId, Phrase and Sentiment. Figure shows the representation of dataset

TABLE I
FIRST FIVE ROWS OF DATASET

	PhraseID	SentenceID	Phrase	Sentiment
0	1	1	A series of escapdes demonstrating the adage...	1
1	2	1	A series of escapdes demonstrating the adage...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

B. 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 Keras framework. So, below is the list of library used:

- Pandas
- Numpy
- Seaborn
- Matplotlib
- Sklearn

C. Understanding the Features

In this paper, I have plot each features as a sub plot which helps to visualize the value of data into it. I have taken first 100 instances of each features and plot it. In order to visualize the graph, I have made use of Matlabplot library which is the most used library for visualization. The comparison of each feature is show in figure 3.

D. Coding

I are 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 5 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.

```
# 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()
```

Listing 1. Preprocessing Steps

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- 0.001 to 0.0000001
- Epochs- 5 to 12
- Batch Size- 128, 64, 32
- Optimizer- Adam, Adadelta, SGD
- Kernel Size- 1

III. PROPOSED MODEL

A. Data Pre-Processing

I have used Rotten Tomatoes movie review dataset and loaded into pandas dataframe. The model includes various libraries such as CSV which is used to import and export format for spreadsheets and databases, Pandas to read our dataset used, Urlib for opening urls, matplotlib is used for plotting library for python programming and its numerical mathematics extension Numpy. Use sklearn for preparing our dataset to train and test the model with. The Dataset is divided in to 70:30 ratio for training and testing purpose. This model performs stemming which is process of removing inflection in words to their root words, Lemmatization which is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item, Stopwords which exclude the words that do not add meningto a sentence such as he, the, is etc, and removal of punctuations which exclude all the punctuation's from the sentences.

```
for l in range(len(documents)):
  label = documents[l][1]
  tmpReview = []
  for w in documents[l][0]:
    newWord = w
    if remove_stopwords and (w in stopwords_en):
      continue
    if removePuncs and (w in punctuations):
      continue
    if useStemming:

      newWord = lancaster.stem(newWord)
    if useLemma:
      newWord = wordnet_lemmatizer.lemmatize(newWord)
    tmpReview.append(newWord)
  documents[l] = (tmpReview, label)
  documents[l] = (' '.join(tmpReview), label)

print(documents[0])
```

Listing 2. Preprocessing Steps

B. Proposed Model

Due to the low accuracy problem faced in pytorch, instead we used keras library and added different layers of CNN are defined namely input layer, pooling layer and output layer. Each has different functionalities. Input layer feeds the input to the network. Pooling layer reduces the number of parameters and computation in the network, controlling over-fitting by progressively reducing the spatial size of the network. The output layer outputs the returns the output of the network.

The feed method is used for feeding the input to the network and processing it through several layers of network. Various activation functions can be used with CNN like Sigmoid, softmax, tanh, ReLU etc. Out of all these, ReLU is used here as it does not activate all the neurons at same time. It increases the computation efficiency by triggering few neurons at a time. A flatten layer is also defined which collapses the spatial dimensions of the input into the channel dimension.

Keras provides various optimizers such as SGD, Adam, Adadelta, etc. for increasing the efficiency of the output. Here AdaDelta optimizer is used for optimizing the output as it gives better performance in comparison to others. A method model loss is also defined which is used for calculation of L1 Loss and accuracy. Thereafter, batch size is chosen which determines the number of batches in which the data will be sent to the model for training the model. After that the number of epoch are chosen. One epoch is when an entire dataset is passed both forward and backward through the neural network only once. At the end of all the epochs, the L1 loss, F1 Score, Precision, Recall and accuracy of the model are calculated.

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3,
                 activation='relu',
                 input_shape=(2000,1)))
model.add(Conv1D(128, kernel_size=5, activation='
    relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Dropout(rate = 0.50))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
# model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Listing 3. Training Model

TABLE II
OPTIMUM HYPER-PARAMETERS FOR PROPOSED MODEL

Hyper-Parameter	Value
Learning Rate	0.5
Optimizer	AdaDelta
Epoch	12
Batch Size	64
Number of Layer	5
Loss Function	L1 Loss

.

C. Experimental Analysis

The main focus is on improving the accuracy of the model. There are various factors that contribute to accuracy such as value of batch size, number of convolutional layers, activation functions used, optimizers used, etc. The number of epochs are kept constant throughout the process and value is set to 3. The batch size is taken as 64. The effect of various activation functions on accuracy and different optimizers are shown in table IV respectively.

TABLE III
DIFFERENT OPTIMIZER

Sr No.	Optimizer	Accuracy	Loss function	F1 Score
1	SGD	0.5877	1.0545	0.2798
2	AdaGrad	0.6054	1.0341	0.39855
3	Adadelta	0.6253	1.0221	0.61489
4	Adam	0.6125	1.0221	0.5463

TABLE IV
COMPARISON OF ACTIVATION FUNCTIONS

Activation function	F1 Score
ReLU	0.615
Softmax	0.623
Sigmoid	0.212

ReLU activation function provided the best results out of all the three activation functions tested.

Considering optimizers, all the optimizers tested yielded almost similar results but among all Adadelta provided the best value of accuracy.

D. 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 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.

E. 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.

```
from keras.models import load_model
model.save('model.h5')
model = load_model('/content/gdrive/My Drive/model.
    h5',
                   custom_objects = {'f1_m': f1_m,
    'precision_m':precision_m,'recall_m' : recall_m
    })
```

Listing 4. Saving and loading model

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 Rotten Tomatoes dataset containing the more than one lakh fifty thousand instances and 4 attributes and two target attributes. Out of the 4 attribute. I

have implemented multi class sentimental analysis for movie reviews . After hyper-parameter turning, I am able to achieve the highest F1 Score of 72 percent for training and 62 percent for testing.

REFERENCES

[1] https://pytorch.org/docs/stable/optim.html
[2] https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing
[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'