Multi-class Sentiment Analysis using CNN

Vikas Trikha
Department of Computer Science
Lakehead University
Thunder Bay, Canada
vtrikha@lakeheadu.ca

***Abstract*—This paper attempts to conduct analysis on rotten tomato data to produce Multi-class Sentiment Analysis applying a simple CNN. The primary data has 156060 entries. It consists of four attributes, which is; PhraseId, SentenceId, Phrase, and Sentiment. In this exercise, we practice Bag of Words (BoW) and term frequency-inverse document frequency (tf-IDF) concerning data vectorization. Vectorized data is then passed to our Convolutional neural network (CNN) models. The data has been divided into 70:30 in which 70 percent is used for training and validation, and 30 percent is used for testing. CNN model has been prepared using several layers like convolutional, dense, dropout, max-pooling, average pooling layers, and works with activation functions like ReLu, sigmoid, and adam to improve the overall accuracy of the model. The model's performance metric parameters are F1 score, precision, accuracy, and recall score.**

***Index Terms*—Multi-class sentiment analysis, convolutional layer, Bag-of-words, Term frequency-inverse document frequency (tf-IDF), max-pooling layer, average-pooling layer, recall, precision, Accuracy, F1 score, dropout layer, dense layer, vectorization**

I. Introduction

According to the given problem statement, we have to implement a robust and scalable convolution layer neural network for text-based movie reviews from the given raw dataset provided by rotten tomatoes using multi-class sentiment analysis. The primary focus is to recognize the underlying statement of a movie review based on its textual information. The dataset train.tsv is tab-spread values, which is divided into train and test set in the ratio of 70:30. The problem statement is solved using various vectorizers like the bag of words, Term frequency-inverse document frequency. For the implementation of the above-discussed model, we used the Keras library to train our model. The output of this assignment can be used as movie recommender. Sentiment analysis is machine learning is profoundly rooted with Natural language processing and text mining. It could be applied to ascertain the attitude of the reviewer with respect to overall polarity of the reviews. OPINION mining (often referred to as Sentiment Analysis) relates to identification and classification of the viewpoint or opinion represented in the text span, practicing information retrieval and computational linguistics. The primary purpose of Opinion mining is to discover the polarity of comments (positive, negative, or neutral) by extracting features and components of the object that have been commented on in several documents.

II. Literature Review

As per the recent writing, a CNN network can be applied in various applications that can be either related to medical purposes or related to financial issues, out of which few can be; cancer prediction or stock market prediction. One of the applications in which machine learning can be further used is doing multi-class sentiment analysis for movie reviews. Lina Zhou et al., [1] studied movie review mining utilizing machine learning and semantic orientation. Supervised classification, including text classification techniques, are adopted in the recommended machine learning program to analyze the movie review. A corpus is formed to describe the data in the documents, and all the classifiers are trained to utilize this corpus. Thus, the proposed technique is more effective. Though the machine learning approach practices supervised learning, the recommended semantic orientation approach practices unsupervised learning because it does not require prior training in order to mine the data. Experimental results showed that the supervised approach achieved 84.49 % accuracy in three-fold cross-validation and 66.27% accuracy on hold-out samples. Bo Pang et al., [2] used machine learning techniques to investigate the effectiveness of classification of documents by overall sentiment. Experiments demonstrated that the machine learning techniques are better than human-produced baseline for sentiment analysis on movie review data. The experimental setup consists of a movie-review corpus with randomly selected 700 positive sentiment and 700 negative sentiment reviews. Features based on unigrams and bigrams are used for classification. Learning methods Nave Bayes, maximum entropy classification, and support vector machines were employed. Inferences made by Pang et al., is that machine learning techniques are better than human baselines for sentiment classification. Whereas the accuracy achieved in sentiment classification is much lower when compared to topic-based categorization. Zhu et al., [3] suggested aspect-based opinion polling from free form textual customer reviews. The aspect of relevant terms employed for aspect identification was learned using a multi-aspect bootstrapping method. A proposed aspect-based segmentation model segments the multi-aspect sentence into single aspect units, which was used for opinion polling. Using an opinion polling algorithm, they tested on real Chinese restaurant reviews achieving 75.5 percent accuracy in aspect-based opinion polling tasks.

III. Dataset

The dataset used for this assignment consists of four data attributes which are Phrase Id, Sentence Id, Phrase, Sentiment. The raw dataset has been acquired from github and the information has been recorded from rotten tomatoes review. Rotten tomato is a movie critique which rates and reviews the movies with proper textual description along with detailed numerical figures. We will use this dataset for multi-class sentiment analysis and would categorize the phrases into 5 classes which are displayed in the below table.

TABLE I
CLASSES FOR MOVIE REVIEWS

Class category	Class Description
4	Very Positive
3	Positive
2	Neutral
1	Negative
0	Very Negative

The dataset is provided in .tsv format which essentially means tab separated values. TSV files are used for raw data and can be imported into and exported from spreadsheet software. TSV files are essentially text files, and the raw data can be viewed by text editors, though they are often used when moving raw data between spreadsheets. The dataset consists of 156060 rows and 5 columns.

IV. Pre-processing

After loading the dataset in our model, the next step is the pre-processing of the data. The evaluation of the designed model majorly relies on how we pre-process the data. Every dataset is different and requires a unique set of pre-processing. In this experiment, we work on a text dataset, so various text pre-processing techniques are incorporated. The first step is to remove all the stopwords, which usually means the commonly used words such as a, an, the, in. The stopwords are removed using nltk library. The second step is to remove punctuations to remove unwanted characters to attain the more normalized review of the phrase. These steps are to be performed to extract the meaningful words and focus on the required concepts to obtain the accurate classification of the terms. Further, it is carried by stemming, which means the process of reducing a word to its word stem and removing its prefixes and suffixes. Stemming is essential in Natural Language Understanding. There are generally two types of stemmers which could be used; Lancaster and Porter. Lancaster is ordinarily more aggressive than a Porter stemmer. For this assignment, we've used the Lancaster stemmer in our deep learning model. The next step after performing stemming is Lemmatization. Lemmatization is the process of grouping together the different inflected forms of a word in order to analyze those words as a single item, Lemmatization is similar to stemming, but it brings context to the words. So it links words with similar meaning to one word. Lemmatization is required for morphological analysis of a word. Hence stemming and Lemmatization is two key essentials in any text pre-processing techniques, and it can be used again with the help of nltk library by importing "WordNet Lemmatizer." Combining everything under one umbrella, Tokenization is a common task in NLP, and that is performing a task of chopping a character into pieces, and those pieces can be referred to as tokens.

V. Proposed Methodology

A plethora of Deep Convolutional neural networks have been proposed and applied in previous researches. A lot of them are dynamic real-time applications like face detection, sentimental analysis, pattern recognition, or prediction of any values. In this research, we have worked upon rotten tomatoes raw dataset downloaded from Github(link) to perform multi-class sentimental analysis using a CNN Model constructed with a couple of deep convolutional layers. The data is pre-processed using tokenization techniques aided by Python's highly recommended library - NLTK. In the early stage of tokenization, the punctuations, stop-words, and whitespaces were discarded, and later phases involved the usage of stemming and lemmatization. The details about pre-processing have been mentioned in the above sections. After bifurcating the dataset into training and testing segments, the training data axes were vectorized using vectorizers such as BOW(Bag of Words) and TF-IDF (Term Frequency Inverse Document Frequency) methods. The process of converting NLP text into numbers is called vectorization. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things: A vocabulary of known words and a measure of the presence of known words whereas TF-IDF weight is a statistical measure employed to assess how relevant a word is to a document in a collection or corpus. The importance progresses proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Refer to the equations below for better understanding about TF-IDF.

$$TF(t) = \frac{Number\ of\ times\ t\ appeared\ in\ a\ document}{Total\ number\ of\ terms\ in\ a\ document}$$

$$IDF(t) = \log_e(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ terms\ t\ in\ it})$$

$$TF - IDFscore = TF * IDF$$

The vectorizers: CountVectorizer and TfidfVectorizer are fetched from Python's feature extraction libraries. CountVectorizer works on Terms Frequency, i.e., counting the occurrences of tokens and building a sparse matrix of documents x tokens. Being a tabular dataset, the sequential DCNN model applied to this dataset is designed with several 1D convolutional layers. The activation function: ReLU is used after convolutional layers, and the model is concluded with a dense layer, followed by the 'Softmax' activation layer.

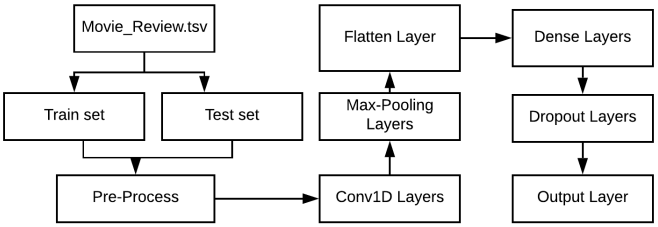

Fig. 1. Overview of Proposed Method

VI. Results & Evaluation

Moving on to the next phase, the developed model was tested on four basic qualitative parameters, which were Accuracy score, F1 score, Precision, and Recall score. Please find below the equations for all the qualitative parameters.

$$Precision = \frac{True\ Positive}{True\ Positive\ +\ FalsePositive}$$

$$Recall = \frac{True\ Positive}{True\ Positive\ +\ False\ Negative}$$

$$F1Score\ =\ 2 \times \frac{Precision\ *\ Recall}{Precision\ +\ Recall}$$

$$Accuracy\ =\ (\frac{Correctly\ Predicted\ class}{Total\ testing\ class}) \times\ 100$$

Initially, the baseline model was trained with adam optimizer, which gave an accuracy of about 61% without re-sampling the code. Since the data is imbalanced and consists of 50.99% sentiment '2' class records, we did some modifications and introduced re-sampling in the dataset and shuffled all the records to improve the accuracy and quantify the uncertainty of the dataset and got following results.

TABLE II
Evaluation matrix using SGD and Adam optimizer

Parameters	Baseline w/o re-sampling	Model with re-sampling after data splitting
Accuracy	61.83	54.81
Precision	63.17	53.67
Recall	60	38
F1 Score	61.24	44.35

Furthermore, the results were experimented with practicing the baseline model with different optimizers and observed the results written in table III.

VII. Conclusion

The problem statement was solved using the Convolutional Neural network to perform the multiclass analysis of the provided raw dataset. Concluding it, the network designed consisted of input and output layers along with 25 other layers to improve the efficiency of the layer with a ReLu activation function. The model was rigorously tested by changing some factors aiming to improve the overall accuracy of the model. The changes have mainly done consisted of changing of optimizers with varied learning rates, adding multiple convolution layers, performing re-sampling of data, and changing batch size, and after all the modifications, the best accuracy achieved was 61.83 % accuracy, and the accuracy decreases by 10 % when the optimizer is replaced with nadam rather adam optimizer. Upon performing the re-sampling and shuffling the data after data splitting, the model's accuracy decreases by 6% accuracy along with other performance parameters.

TABLE III
Effect of changing optimizers on CNN model

Parameters	adam	nadam
Accuracy	61.83	51.23
Precision	63.17	52.03
Recall	60	51
F1 Score	61.24	51.23

Appendix

```
#defining performance metric parameters
from keras import backend as K
def recall_m(y_true, y_pred):
  true_positives = K.sum(K.round(K.clip(y_true*
    y_pred, 0, 1)))
  possible_positives = K.sum(K.round(K.clip(y_true,
    0 ,1)))
  recall = true_positives / (possible_positives + K.
    epsilon())
  return recall

def precision_m(y_true, y_pred):
  true_positives = K.sum(K.round(K.clip(y_true*
    y_pred, 0, 1)))
  predicted_positives = K.sum(K.round(K.clip(y_pred,
     0 ,1)))
  precision = true_positives / (predicted_positives
    + K.epsilon())
  return precision

def f1_m(y_true, y_pred):
  precision = precision_m(y_true, y_pred)
  recall = recall_m(y_true, y_pred)
  return 2*((precision*recall)/(precision+recall+K.
    epsilon()))

  #Model
  model = Sequential()
model.add(Conv1D(filters=64, kernel_size=2,
    activation='relu', input_shape=(x_train_np.shape
    [1],x_train_np.shape[2])))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=128, kernel_size=2,
    activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=256, kernel_size=2,
    activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=512, kernel_size=2,
    activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=1024, kernel_size=2,
    activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(BatchNormalization())
```

```
model.add(Flatten())
#model.add(Dense(50, input_dim=2, activation='relu')
    )
model.add(Dense(512))
model.add(Dropout(0.8))
model.add(Activation('relu'))
model.add(Dense(256))
model.add(Dropout(0.8))
model.add(Activation('relu'))
model.add(Dense(128))
model.add(Dropout(0.8))
model.add(Activation('relu'))
model.add(Dense(64))
model.add(Dropout(0.8))
model.add(Activation('relu'))
model.add(Dense(5))
model.add(Activation('softmax'))
#model.add(Activation('softplus'))
model.summary()
#opt=adam(lr=0.001, decay= 0.01)
model.compile(optimizer='adam', loss='
    categorical_crossentropy', metrics=['acc',f1_m,
    precision_m, recall_m])
```

Listing 1. Code for CNN Regressor and computing model loss and training data

REFERENCES

[1] "Lina Zhou, Pimwadee Chaovalit, Movie Review Mining: a Comparison between Supervised and Unsupervised Classification Approaches, Proceedings of the 38th Hawaii International Conference on system sciences, 2005.

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up? Sentiment classification using machine learning techniques, In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7986, 2002

[3] Zhu, Jingbo Wang, Huizhen Zhu, Muhua Tsou, Benjamin K. Ma, Matthew, Aspect-Based Opinion Polling from Customer Reviews, IEEE Transactions on Affective Computing, Volume: 2,Issue:1 On page(s): 37. Jan-June 2011.

[4] Jason Brownlee "ReLu activation function". machinelearningmastery.com. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (Accessed Feb 11, 2020)

[5] Jiwen Jeong "CNN". towardsdatascience.com. https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480 (Accessed Feb 11, 2020)