Multi-Class Sentiment Analysis using Deep Learning

Jugal Shah
Department of Computer Science
Lakehead University
Thunder Bay, Canada
jshah5@lakeheadu.ca

***Abstract*—Sentiment analysis is a mathematical approach of extracting and classifying the emotions for a given structured, semi-structured, or unstructured data. A general Sentiment analysis model tries to assign a polarity label, e.g., positive or negative, to the given text or document. Natural language processing (NLP) plays a vital role in building these models. NLP algorithms like bag of words (BoW), term frequency-inverse document frequency (TF-IDF) extract features from the given document or text, and output a mathematical vector, which can then be used to train machine learning models and neural networks. Today with recent development in deep learning models like convolution neural network (CNN) and NLP techniques like Word2vec, better text analysis and classification are achievable, thereby increasing the accuracy of the overall model. CNN represents a class of deep neural networks that are well known for the classification problem, majorly in the domain of image classification and text classification. This paper focuses on the implementation of the CNN model for fine-grained sentiment analysis. Rotten Tomatoes movie review dataset is used as a corpus for building a multi-class sentiment analysis model. The aim of the study to assign sentiment between zero to four for the given phrase of a movie review.**

***Index Terms*—CNN, kernel, learning rate, F1 score, recall, precision, epoch, SGD, Adam**

I. Introduction

Sentiment analysis is extensively used to analyze the opinion of people on online platforms like Facebook, Twitter, IMDB, Rotten Tomatoes. Today with opinions and reviews being posted online more frequently than ever, data overload has become a significant issue. This voluminous data present in natural human language makes it challenging to classify the performance of a product in a market or people's views about a topic. Therefore the need to represent the vast data in statistical form has arisen. With sentiment analysis and NLP, it is possible to take advantage of the text data available to train a deep learning model that can classify the people reviews and opinions into categories making further processing easy.

Today entertainment is a significant part of people's lifestyles; consequently, more and more movies are released each year. With the increasing film count, the movie reviews platforms have gained considerable importance. One such platform is Rotten tomatoes, being one of the most well know movie review websites, in this study we are using rotten tomatoes movie review dataset created by Pang and Lee to do multi-class sentiment analysis [1].

In sentiment analysis, people reviews can be analyzed using a lexical based approach or machine learning-based approach of sentiment analysis. The lexical approach makes use of a dictionary of sentiment polarity such as SentiWordnet to perform sentiment analysis. In contrast, machine learning is a domain specific approach that uses a statistical interpretation of the text vector representation to predict the sentiment using the dataset defined label [2]. Initial researches showcase the application of machine learning for sentiment analysis. This paper follows a deep learning approach proposing a CNN model to perform sentiment analysis on the specific domain of movie reviews. As part of the data preprocessing movie review, the dataset is cleaned using python NLTK library, and word embeddings are performed using the sklearn library. The embedded vectors are then fed to the one-dimension CNN build using Keras library. Google Colab is used as the execution environment for study. Experimenting with different hyperparameters, preprocessing steps and vectorization methods, the paper proposes a model with accuraccy of 61 percent, F1 score 56 percent, precision 66 percent and recall of 51 percent

II. Background

A substantial amount of research has been carried out on sentiment analysis using a diverse set of datasets. Rasika and Prof. Thakare, 2016, outlines an instance of lexical based sentiment analysis on the Times of India movie review dataset [3]. The paper finds the polarity of the movie reviews using SentiWordNet. The dataset is pre-processed by tokenizing the text and removal of stopwords. Each word in the review is then tagged with parts of speech (POS) tag. Post pre-processing features such as positive sentiment words, negative sentiment words, unigram models are extracted to identify the polarity of the text document. Additionally, to improve the performance of the classifier, techniques like Information Gain and Gain Ratio are used to reduce the feature dimension. Random Forest classifier, when fed the reduced feature vectors, achieves an accuracy of 90 percent.

Researchers additionally focus on the machine learning approach to perform sentiment analysis on the given text. Traditional machine learning techniques like Naive Bayes, maximum entropy classification, Support Vector Machine (SVM) have proven to be useful in sentiment analysis; however, deep learning models transcend in understanding complex inputs

according to Jeyaprakash et al., 2019 [4]. The authors propose a CNN model build using Apache MXNet for the Standford movie review dataset. Here words are embedded using GloVe and fed to the CNN model. The model is trained for three epochs with Adam optimizer with a learning rate of 0.01. Overfitting is avoided with the addition of the dropout layer in the model structure. The experiment performs the training and testing of the model with different kernel sizes, achieving around 80 percent accuracy and states that the CNN model does not have a significant effect on classification accuracy. To overcome the limitation of CNN, researchers have started focusing on the application of advance deep learning models like Long short-term memory (LSTM). Aswathi and Lakshmi, 2019, showcase the result of applying hybrid model LSTM-CNN and CNN-LSTM for sentiment analysis on movie reviews [5]. IMDB movie review dataset that consists of positive movie reviews and negative movie reviews are used as the dataset for the study. The proposed architecture mention in the paper involves the preliminary step of data cleaning and input vector generation using the embedding layer. The input vector is they passed to the LSTM layer, which is essential in capturing long term dependencies when working on long texts. Next, the CNN layer receives the output from the LSTM to perform convolution operations and extract features from the input on which the rectified linear unit (ReLU) activation function is applied. The dimension of the features is reduced using the MaxPooling layer and passed to the classification layer created using Keras Dense Layer. The LSTM-CNN model gave an accuracy of 79 percent, whereas CNN-LSTM gave an efficiency of 51.5 percent. The performance analysis is done using learning curves on the training and validation dataset. Creating a vector representation of the text data has been a significant part of the NLP sentiment analysis task. Fan Xian et al., 2016, examines the effectiveness of word vector representation on sentiment analysis [6]. The study applies the Google Word2Vec model for sentiment analysis on APP reviews and achieves F1 of 85.77 percent, recall of 85.20 percent and an accuracy of 86.35 percent.

III. Dataset

In this study, the Rotten Tomatoes movie review dataset is used for sentiment analysis. The dimension of the dataset is 156060×4. Pandas library is utilized for reading the dataset. Table 1 below shows the head of the dataset.

Sentences from reviews are parsed into many phrases, and each phrase is stored as an entry in the dataset. The SentenceId and PhraseId identify the sentence and phrases in the dataset. Each phrase in the dataset is associated with a sentiment label between 0 and 4. Table 2 below represents the sentiment linked to each label. The dataset also suffers of class imbalance, as indicated by Fig. 1.

IV. Data Preprocessing

At the first stage, the dataset is first checked for null values and then partitioned into training and testing using the train and test split functionality from sklearn library. The ratio of training to testing data is 70:30. A random state of 2003 is set so that during each execution, the training and testing data is not modified. Post data splitting below steps are followed for train and test data independently.

TABLE I
Dataset Head

PhraseId	SentenceId	Phrase	Sentiment
1	1	A series of escapades demonstrating the adage that what is good for the goose is also good for the gander , some of which occasionally amuses but none of which amounts to much of a story.	1
2	1	A series of escapades demonstrating the adage that what is good for the goose	2
3	1	A series	2
4	1	A	2
5	1	series	2

TABLE II
Sentiment Labels

Sentiment labels	Sentiment
0	negative
1	somewhat negative
2	neutral
3	somewhat positive
4	positive

A. Data Cleaning

In data cleaning, each phrase in the dataset is converted to lower case and is tokenized using the NLTK word tokenize library. Stop words and punctuations are removed from the phrases. The English stopwords list defined by the NLTK corpus library is used for the removal of stopwords. The actual code snippet for data cleaning is given in the Appendix.

B. Data Lemmatization

Lemmatization is the process of reducing a word to its root form.In this paper, lemmatization is performed on the dataset phrases using the WordNetLemmatizer package of the NLTK

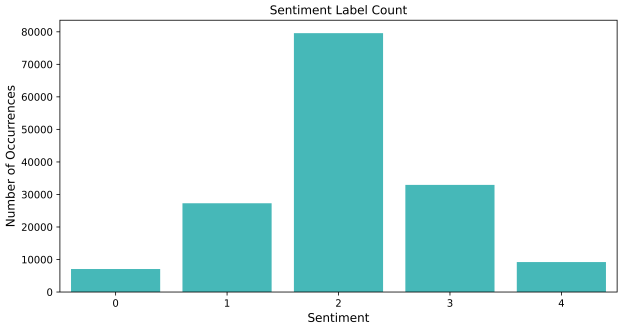

Fig. 1. Sentiment Label Count.

library. The reasoning behind choosing lemmatization over stemming is because it produces meaningful root words for the given word.

C. Data Vectorization

Data vectorization can be done using a variety of NLP techniques. Some of the well-known techniques are BoW, TF-IDF, and Word2Vec. In BoW, the frequently occurring words are given more importance. It forms a matrix of documents against tokens, each cell storing the count of the token in the document. This count is defined as Term Frequency (TF). In TF-IDF, the relevant words are given more importance than the frequent words. Equation (1) denotes the TF-IDF formula. In the formula *t* denotes the terms; *d* denotes each document; *D* denotes the collection of documents.

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{1}$$

Equation (2) shows the expansion of *idf(t,D)*

$$idf(t, D) = log\frac{|D|}{1 + |\{d \in D : t \in d\}|} \tag{2}$$

The numerator in the above equation represents the number of documents in the corpus and denominator represents the number of documents in which the term t occurs. Word2Vec is distinct from the previous two vectorization technique. BoW and TF-IDF perform frequency-based vectorization, whereas Word2Vec is two layers neural network that can store the context of the given text with the help of weights. The output of the Word2vec model can have several dimensions. In this research, we experimented with BoW and TF-IDF; the model performs almost similar under both the algorithms.

D. Data Reshaping

At last the Training and testing data is converted to a NumPy array using the NumPy library. The NumPy array of the phrases are then reshaped to include the third dimension, and the Numpy array of the sentiment labels are converted to categorical to support multi-class classification.

V. Proposed Model

The paper proposes a CNN model to perform sentiment analysis on the movie reviews dataset. Fig. 2 below shows the complete architecture of the CNN model built using Keras library. The training and testing data are first converted into batches of fixed size before passing to the model. The training data is further divided into training data and validation data to keep a check if the model starts overfitting after some number of epochs. Twenty percent of the original training data is kept aside for validation. The batched are first forwarded to the convolution layer of the model that runs thirty-two filters of kernel size two over the data to generate features. The activation function Rectified Linear Unit (ReLU) is applied to the produced features, post which the features are passed to the max pooling layer for downsampling. Additionally, a dropout rate of 0.2 is applied to the neurons, indicating deactivation of 20 percent of neurons to avoid overfitting. The output produced is again forwarded to a set of convolution layer with 64 filters, activation function, and max pooling layer with dropout applied again. The final output is flattened into a single vector, passed to a Dense layer. Softmax activation function is utilized on the Dense layer output to predict multiple mutually exclusive sentiments. The proposed model is evaluated using categorical crossentropy loss, accuracy, precision, recall, and F1 score. Furthermore, the training and validation loss over the epochs were plotted to get the optimal count of epochs before the model starts to overfit.

A. Parameters Configurations

During the research, different permutations and combination of the hyper parameters, model parameters, and model layers were tried and tested. The proposed model performs the best with the configurations shown in Table 3 below. In the study, we experimented with different kernel sizes, like 1,2,3,4,5. The performance of the model was not significantly affected by the kernel size; however, it seems to perform slightly better on kernel size 2.

TABLE III
Proposed Model Parameters Configurations

Parameter	Configurations
Kernel	2
Learning Rate	0.001
Batch size	64
Epoch	25
Optimizer	Adam
Activation function	ReLU and softmax

B. Model Implementation

From a practical perspective, the model is defined using python language, with each layer imported from the Keras library. Keras sequential model is used to stack different layers mention in the model architecture. Before training the model, the learning process is configured using the compile method. Once compiled, the model is trained using the fit method, passing the input data and label for training along with epochs, validation split, and the batch size. Verbose is activated during training to archive the epoch results. The model code snippet is mention in the Appendix.

C. Model Evaluation

The proposed model is evaluated using categorical cross-entropy loss, accuracy, precision, recall, and F1 score. Table 4 below shows the model performance metrics.

TABLE IV
Proposed Model Performance Metrics

Metric	Value
Categorical Cross Entropy Loss	1.005
Accuracy	0.611
F1 Score	0.581
Precision	0.665
Recall	0.518

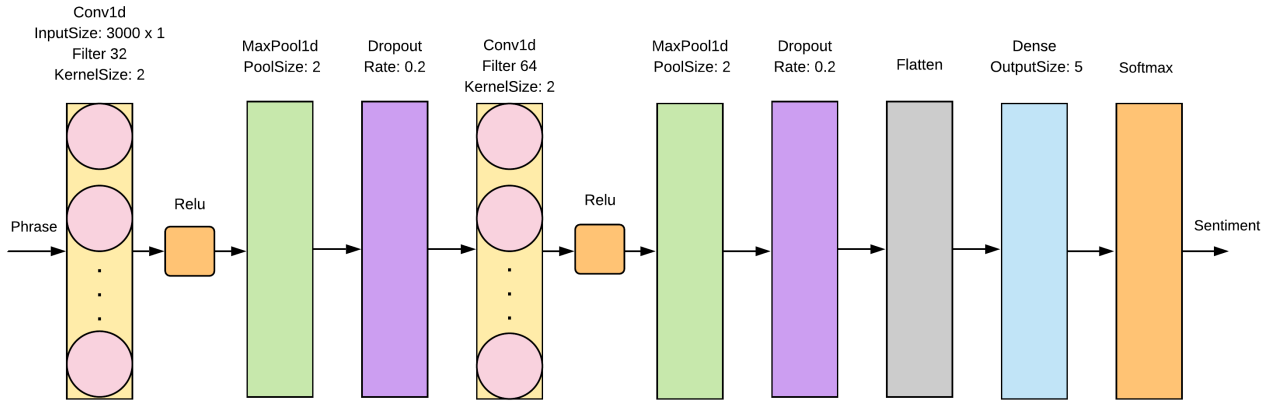

Fig. 2. Proposed CNN Architecture.

Keras backend library is used to calculate categorical cross-entropy and accuracy; however, the recall precision and F1 score are calculated using the equations below.

$$Recall = \frac{truepositives}{truepositives + falsenegatives} \tag{3}$$

$$Precision = \frac{truepositives}{truepositives + falsepositives} \tag{4}$$

$$F1Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{5}$$

Equation (6) denotes the general equation of accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

In the equation TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative.

D. Model Overfitting

The validation dataset is used during training to avoid overfitting. Fig. 2 and Fig. 3 shows the accuracy and loss of the training data and validation data over twenty-five epochs. It can be clearly seen in figures that the validation data loss is steadily decreasing with training loss, with few spikes and the accuracy of both data steadily increasing,

VI. Experiment Analysis

During the experimentation part of the research, in addition to the proposed model, two more additional models were built.

A. Resampled Model

In this model, once the data is split into training and testing, to handle the class imbalance, oversampling is performed on the train data only, using the resample functionality from the sklearn library.For the resampled model, in 25 epochs, the training and validation accuracy as well as F1 score come round 60 percent with no overfitting indicated by the graph; however, the testing accuracy falls to 54 percent with an F1

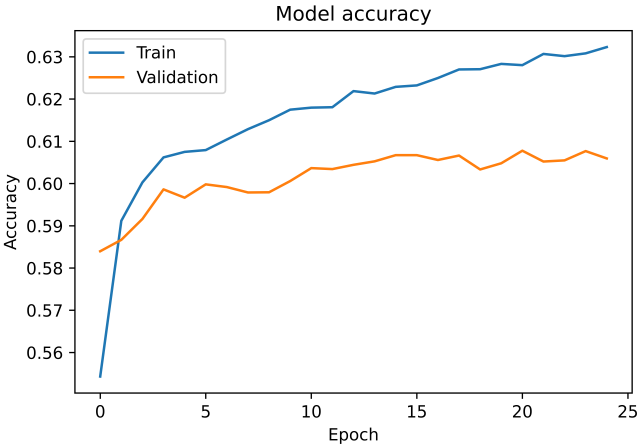

Fig. 3. Training and Validation Accuracy.

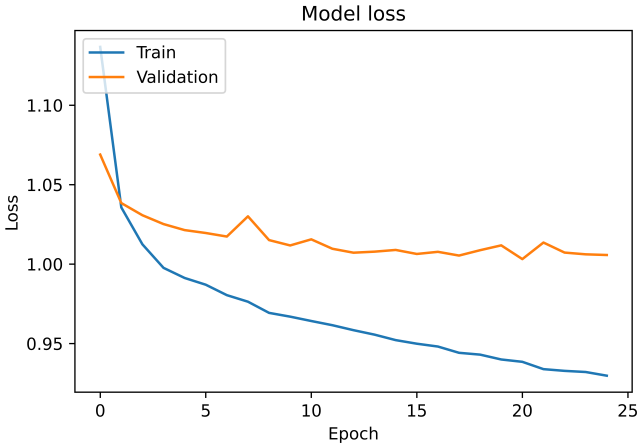

Fig. 4. Training and Validation Loss.

score of 49 percent. The reason for this behavior is to be researched.

B. Embedding Layer Model

In this model, in addition to the resampling of train data, alternately of using TF-IDF, the training data is embedded using Tokenizer API provided with Keras. Tokenizer helps in representing each word in the phrase as a unique number. Padding is applied to vectors generated by the tokenizer to bring all vector to the same length. While building the model Embedding layer defined by Keras is added as the first layer. It takes three inputs; first is the number of unique words in the train data, second is the embedding dimension, which is set to 100, and lastly, the max length of a review.With embedding, the models training and validation accuracy as well as F1 score come round 80 percent with no overfitting indicated by the graph; however, the testing accuracy falls to 60 percent with an F1 score of 59 percent. The reason for this behavior is to be researched.

VII. Conclusion

This paper presents the application of one-dimensional convolution neural networks to perform sentiment analysis on movie reviews. The study highlights the fact that experimentation with data preprocessing and text vectorization can offer notable outcomes. The predefined Word2vec model gives us another area to explore to improve model performance. The proposed model was a result of rigorous trials that revolved around working with numerous combinations of hyperparameters and the organization of the CNN layers. The additional model mentioned under experiment analysis shows promising results on training and validation results and can be used as a good starting point to improve the performance of the model further.

Appendix

A. Data Cleaning

```
#utility function to clean the dataset
def datacleaning(remove_stopwords,useStemming,
    useLemma,removePuncs,newdata):
  cleanReview=[]
  for x in range(0,len(newdata.values)):
    tmpReview=[]
    for w in nltk.word_tokenize(newdata.values[x]):
        newWord = str(w).lower()
        if remove_stopwords and (w in stopwords_en):
        #if stopword remove
            continue
        if removePuncs and (w in punctuations):
        #if the word a punctuation remove
            continue
        if useStemming:
            #if useStemming is set to True
            newWord = lancaster.stem(newWord)
        if useLemma:
            newWord = wordnet_lemmatizer.lemmatize(
    newWord)
        tmpReview.append(newWord)
    cleanReview.append(' '.join(tmpReview))
  return cleanReview
```

Listing 1. Data Cleaning

B. CNN Model

```
#Model Defination

cnnmodel = Sequential()
cnnmodel.add(Conv1D(filters=32, kernel_size=2,
    activation='relu',input_shape=(x_train_np.shape
    [1],x_train_np.shape[2])))
cnnmodel.add(MaxPooling1D(pool_size=2))
cnnmodel.add(Dropout(rate=0.2))
cnnmodel.add(Conv1D(filters=64, kernel_size=2,
    activation='relu'))
cnnmodel.add(MaxPooling1D(pool_size=2))
cnnmodel.add(Dropout(rate=0.2))
cnnmodel.add(Flatten())
cnnmodel.add(layers.Dense(5, activation='softmax'))

#Compile the model
cnnmodel.compile(optimizer='adam',loss='
    categorical_crossentropy', metrics=['accuracy',
    f1_m,precision_m,recall_m])
cnnmodel.summary()

#Train the model
modelhistory= cnnmodel.fit(x_train_np, y_train_np,
    epochs=25, verbose=1, validation_split=0.2,
    batch_size = 64)
```

Listing 2. CNN Model

References

[1] Pang, Bo, and Lillian Lee. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales." In Proceedings of the 43rd annual meeting on association for computational linguistics, pp. 115-124. Association for Computational Linguistics, 2005.

[2] 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.

[3] Wankhede Rasika, and A. N. Thakare. "Design approach for accuracy in movies reviews using sentiment analysis." In 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), vol. 1, pp. 6-11. IEEE, 2017.

[4] Chelladurai Jeyaprakash, Suresh R. Chelladurai, and Biju R. Bajracharya. "Accuracy of Convolution Neural Networks for Classifying Sentiments on Movie Reviews." In 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 969-972. IEEE, 2019.

[5] Sajeevan Aswathi, and K. S. Lakshmi. "An enhanced approach for movie review analysis using deep learning techniques." In 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 1788-1794. IEEE, 2019.

[6] Fan Xian, Xiaoge Li, Feihong Du, Xin Li, and Mian Wei. "Apply word vectors for sentiment analysis of APP reviews." In 2016 3rd International Conference on Systems and Informatics (ICSAI), pp. 1062-1066. IEEE, 2016.