Sentiment Analysis of Movie Review with CNN

Ngudup Tsering
1104377
MSc Computer Science
Lakehead University, Thunder Bay

***Abstract*—With a large number of movies, tv shows and other short drama being released every year, and with the access to such movies getting easier with services like Netflix, Amazon, Apple TV etc, the movie review is becoming more relevant in offering insight to make an informed choice of utilizing the capital for entertainment. One of the most trusted movie review websites is Rotten Tomatoes. It recommends a program based on their Tomatometer. Tomatometer represents the percentage of professional critic reviews that are positive or negative for a given film or television show. Fresh tomatoes represent positive sentiment, whereas rotten reviews mean that the movie has negative sentiments. Therefore, to extract the polarity of the reviews using Natural Languages Processing (NLP) methods, this paper proposes a new model to predict the sentiment of the movie on the rotten tomatoes using 1D Convolutional Neural Network (CNN) after Term Frequency - Inverse Document Frequency (TF-IDF) vectorization. The result of the experiment shows that the proposed method offers 62.37 per cent of accuracy on the benchmark dataset of Rotten Tomato.**

***Index Terms*—NLP, CNN, Lemmatization, TF-IDF**

I. Introduction

Sentiment analysis is employed extensively in extracting relevant polarity from text reviews using NLP methods and tries to get the fundamental perspective of the content, which can be something that carries a subjective assumption. With the sharp rise in internet usage and subsequent online marketing, rating and reviews from users affect buying trends and reputation of companies. As mining user reviews is very time consuming and long process for the nature of reviews are unstructured, therefore techniques of sentiment analysis play an important role. Sentiment analysis examines the opinion of people on the generally used product and services and finds whether the given opinion is positive, neutral or negative.

The movie reviews are generally in the form of text and not structured in nature. Therefore, these stop words and other undesirable data are expelled from the reviews for further investigation. This process in sentiment analysis is called preprocessing of the given document i.e. tokenization, stop word removal, stemming or lemmatization, and finally classification. Tokenization identifies the basic units by separating texts into sentences and words. Stemming and lemmatization are processes of removing prefixes and affixes to convert the word into its stem or root words. Finally, the classification is labelling the test data based on the previously trained data with a classification model. Distinctive parameters are then used to assess the execution of the machine learning calculations
This paper explores the CNN architecture using 1D CNN model to achieve the result. Many measures can be implemented to evaluate the model, however the paper will focus on F1 score, accuracy, precision and recall.

II. Literature Review

Many research has been done in the area of sentiment analysis on reviews. The dataset used for this assignment was first collected from Rotten Tomato by Bo Pang et al. [1]. They addressed the rating-inference problem from a binary review as "thumbs up" or "thumbs down", to a multi-point scale (e.g., one to five "stars") system. They begun with evaluating human performance at the task and then went on to apply a meta-algorithm, based on a metric labelling formulation of the problem, that alters a given n-ary classifier's output in an explicit attempt to ensure that similar items receive similar labels. When the novel similarity measure appropriate to the problem was employed, a significant improvement over both multiclass and regression versions of Support Vector Machines (SVM) was found. Socher et al. [2] have inspired a Kaggle competition on the same dataset, where the authors propose the use of a Recursive Neural Network (RNN) which builds on top of grammatical structures. Furthermore, they create a dataset called Sentiment Treebank, which contains sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences. With their technique on this dataset, the authors reported a 5% improvement in the accuracy of sentiment prediction compared with the current state-of-the-art. Abhilash et al [3] explore a similar problem with Amazon reviews using Amason API and exploits Support Vector Classifier (SVC) and Naive Bayes (NB) for classification of online reviews using a web model using supervised learning methods. They used unigram and weighted unigram s to train their machine learning classifiers and discovered that it works well on weighted unigram with SVM provided the best result. Similar researches were also carried with other datasets of IMDB review.

All approaches that exist either use SVC, RNN, Long Short Term Memory (LSTM) or Naive Bayes model to solve this sentiment analysis . This proposal is an attempt to look at the multicalass classification from 1D CNN [5] approach and offer a fresh perspective on the novelty of 1D CNN and it's capacity to tackle it.

III. Methodology

A. Dataset

The dataset contains tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been

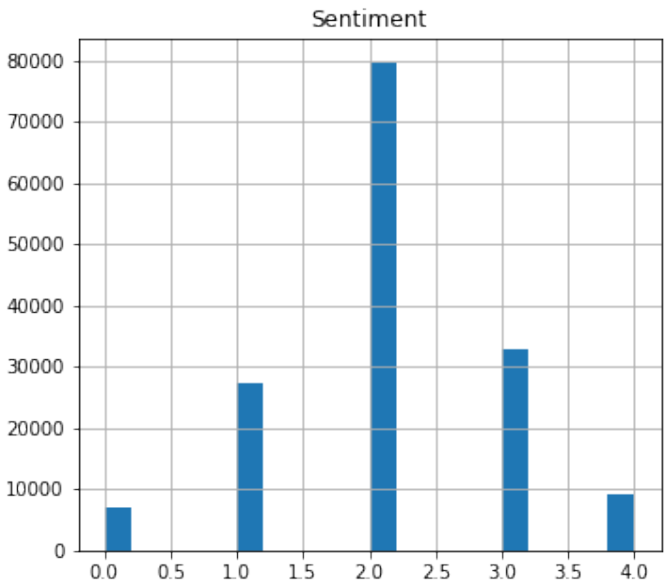

Fig. 1. Dataset Visualisation.

preserved for the purposes of benchmarking and therefore test dataset is not available with their respective sentiments. The sentences have been shuffled from their original order. Every Sentence is parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. The repeated phrases (such as short/common words) are only included once in the data. train.tsv contains the phrases and their associated sentiment labels. Table 1 show the sentiment and their respective labels

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

TABLE I
SENTIMENT LABELS.

Fig. 1 shows a distribution of sentiments of the whole dataset. There is total of 156060 phrases. The most number of sentiment in the distribution is neutral with 80k phrases. However, both extreme ends of the sentiments are relatively smaller under 10k phrases. Huge imbalance in the data usually has an impact on the effectiveness of the model and needs some balancing. Data augmentation in sentiment analysis to push up the phrase counts of low sentiment counts is usually discouraged as it has tendencies to skew the whole dataset towards prejudice of the source of augmentation. Therefore to provide equal representation in train and test split, stratified spliting is implemented.

B. Preprocessing

The preprocessing of text is an important step in the processing of NLP tasks. It transforms text into a more manageable form, so that machine learning algorithm can perform better.

1) Tokenization: This step splits or segments a text into tokens which are words, numbers, and symbols. Sentence tokenization is carried with NLTK library which is trained on English language.

2) Stop words and Punctuation Removal: After tokenization, unimportant or insignificant words that won't affect the classification process are removed. The steps involve removal of stop words and punctuation. Stop words are those words that appear frequently in a sentence or review that does not contribute to the process of classification.

3) Stemming: The Porter stemming algorithm (or 'Porter stemmer') [4] is a process for removing the commoner morphological and inflexional endings from words in English. Its main purpose is as part of a term normalisation process.

4) Lemmatization: Lemmatize is carried using WordNet's built-in morphy function that returns the root word when the the input word is found in WordNet, otherwise it remain unchanged. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

C. Vectorization

It is the general process of turning a collection of text documents into numerical feature vectors.

1) TF-IDF: While Bag of Words (BoW) and n-gram modelling methods concentrated more on higher frequency parts of the review, they conveniently ignored the portions which might be less frequent but have more significance for the overall polarity of the review. To account for this, we created feature representations of words using TF-IDF. The feature representation for this model is similar to the Bag of Words model except that we used TF-IDF values for each word instead of their frequency counts. To limit the number of words common to both positive and negative reviews, we ignored all the words whose count was more than 50 as they would not contribute much to the classifier.

$$\omega_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{1}$$

where $\omega_{i,j}$ is weight for token i in document j,
$tf_{i,j}$ is number of occurrences of token i in document j,
df_i is number of documents that has token i, and
N is total number of docs in the training corpus.

D. Proposed Network Architecture

A CNN recognises simple patterns within the data, which can then be used in higher layers to shape more complex patterns. When the position of the feature within the segment is not of great significance, a 1D CNN which is powerful in extracting interesting features of the dataset from its shorter

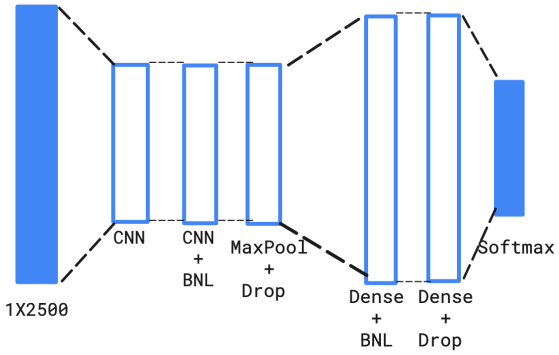

Fig. 2. Network Architecture.

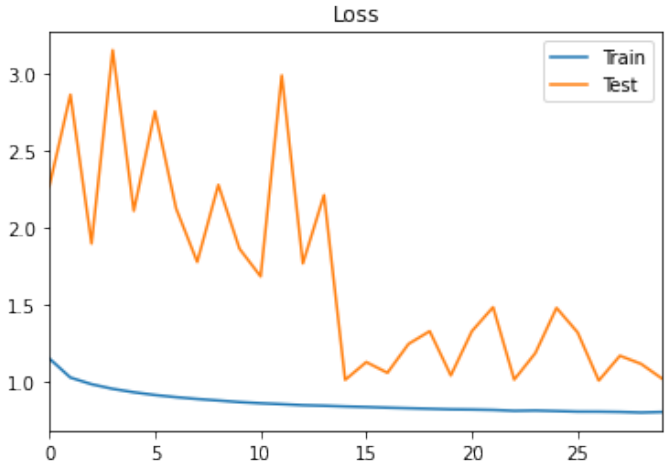

Fig. 3. Loss v/s Epoch.

segments, can be used efficiently. Hence, it can be applied in the analysis of time sequences of sensor data and Natural Language Processing. Fig. 2 succinctly represents the network architecture adopted for the assignment with the following components.

- **Input Layer(1D CNN) :** The input data has been preprocessed through tokenization, stop words and punctuation removal, and lemmatization. Every instance of the input has 2500 features by limiting the vectorized maximum feature. It defines a 64 kernel of size 1. One filter would allow the neural network to learn a single feature in the first layer which is not adequate, hence 64 filters are defined to detect 64 features.
- **1D CNN layer :** The second layer defines a 128 kernel of size 1. To extract more complex feature, more depth is created with subsequent layers.
- **Batch Normalisation:** Batch Normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.
- **Pooling layer:** A Pooling Layer is usually used after a CNN layer in order to subsample the input to provide a simpler output that helps prevent overfitting of the data. The proposed model uses max-pooling layer after two consecutive CNN layers.
- **ReLU Activation Layer:** The activation function is responsible for the transformation of the combined weighted input from the node into the ouput node. The rectified linear activation function is a piecewise linear function that will output the input directly if is positive, otherwise, it will output zero. For many neural networks, ReLU has become the defacto activation function, because models that use ReLU are easier to train and often are better at the performance.
- **Fully Connected (FC) Layers:** The FC layers use the flattened output from the CNN layers in the vector of height 256. The proposed model has two FC layers and the last FC layer is responsible for providing a five output to predict five different sentiment. The last FC layer take a softmax as a classifier for it's a multiclass classifier.

A total of 40,970,757 hyperparameters is learned during training epochs for the proposed model. Different combination of network architecture was tried which necessitated the need to add the Batch Normalisation layers and Dropout layers. Without these regularisation techniques, the network was overfitting, and therefore wasn't reliable.

GitHub link to the said model is https://github.com/NTsering/Sentiment-Analysis.git

IV. Performance Evaluation:

A. Categorical Crossentropy Loss

Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. No data point can belong to more than one class.

Categorical crossentropy compares the distribution of the predictions with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. A class is represented as a one-hot encoded vector, and the closer the model's outputs are to that vector, the lower the loss.

$$L(y, \widehat{y}) = \sum_{0}^{N} \sum_{0}^{M} (y_{i,j} * \log(\widehat{y_{i,j}})) \tag{2}$$

where $\widehat{y}$ is the predicted expected value and y is the observed value.

B. Accuracy

This calculates the accuracy of a single (y, $\widehat{y}$) pair by checking if the predicted class is the same as the true class. It does this so comparing the index of the highest scoring class in $\widehat{y}$ vector and the index of the actual class in the y vector.

$$Accuracy = \frac{Total\ Correct\ Predictions}{Total\ Predictions} \tag{3}$$

Model	Train					Test				
Metrics	**Accuracy**	**Loss**	**Precision**	**Recall**	**F-measure**	**Accuracy**	**Loss**	**Precion**	**Recall**	**F-measure**
1D CNN	0.692	0.798	0.726	0.641	0.681	0.623	1.018	0.582	0.646	0.612
Bernouli Naive Bayes	0.61	-	-	-	-	0.59	-	-	-	-

TABLE II
RESULTS AND COMPARISON.

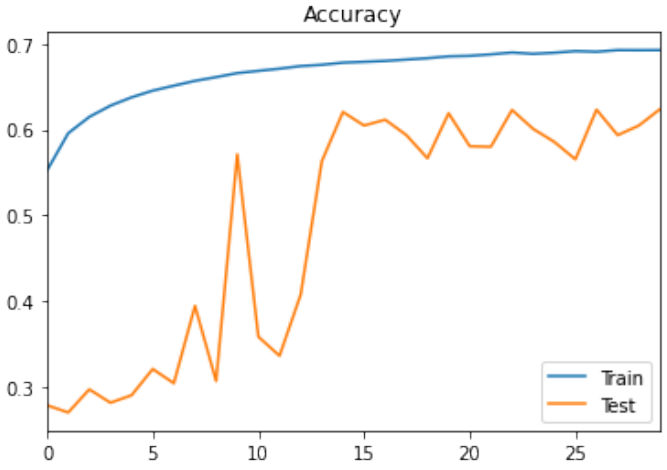

Fig. 4. Accuracy v/s Epoch.

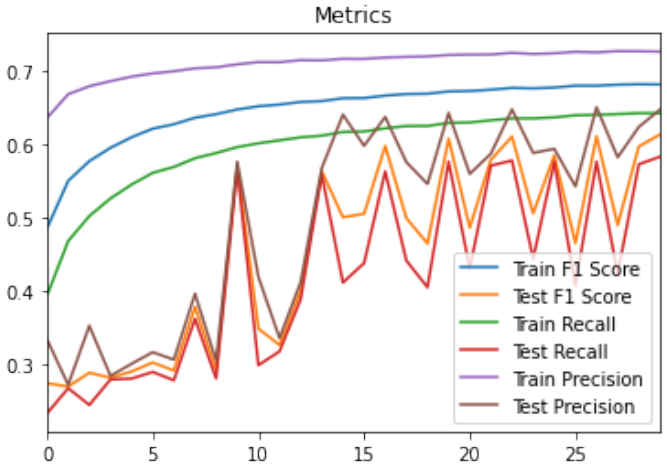

Fig. 5. Other Metrics v/s Epoch.

C. Precision

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances.

$$Precision = \frac{\sum True\ Positive}{\sum Predicted\ condition\ positive} = \frac{\sum True\ Positive}{\sum (True\ Positive + False\ Positive)} \quad (4)$$

D. Recall

Recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved.

$$Precision = \frac{\sum True\ Positive}{\sum Condition\ positive} = \frac{\sum True\ Positive}{\sum (True\ Positive + True\ Negative)} \quad (5)$$

E. F1 Score

A measure that combines precision and recall is the harmonic mean of precision and recall. It considers both the precision and the recall of the test to compute the score, and therefore provide a better measure of the model, especially when the data is imbalanced.

$$F1Score = 2 \times \frac{Precision\ \times\ Recall}{Precision\ +\ Recall} \quad (6)$$

The *Precision* and *Recall* is calculated against the rest of the class and averaging them during the calculation.

V. RESULTS

Based on the results shown in TABLE II of the proposed model, it is evident that our model using 1D CNN is capable of sentiment analysis of movie review. The accuracy of the model is 62.37% with F1 Score of 61.21%. The performance of the model is increased and the Loss decreased with successive training cycles as shown in Fig. 3 and 4. Fig. 5 presents an holistic representation of the variation of other metrics such as F1-Score, Precision, and Recall over the epoch.

The highest score on this dataset so far is achieved by Mark Archer on the Kaggle competition with 76.5%. However, the test benchmark is different for the competition. Furthermore, no details on their model of implementation is available for general public. Usually such problem is either solved using advanced sequence handling components such as RNN or LSTM, or tradition machine learning approach such as Naive Bayes, Random Forest, or SVM. It is prudent to acknowledge that CNN model provides better score than Bernoulli Naive Bayes model with same training input.

VI. CONCLUSION

This paper provides a fresh approach to classification using 1D CNN for sentiment analysis of movie review. Further experiments and fine-tuning will improve the model incrementally. Additionally, it evinces the capacity of CNN in

classification and text analysis which is virtually confined to image and video-related applications. Furthermore, the different score metrics of the proposed model is promising and gives better score than Bernoulli Naive Bayes model with same training input and testing set.

CNN architecture, like Artificail Neural Network (ANN), is prone to overfitting. The regularisation layers included in the proposed model proved effective in arresting the overfitting factor. Further improvement can also be done with regard to dataset as the existing dataset imbalance as depicted in Figure 1 and therefore the extreme end of the sentiments are not represent in proportion to the rest of the data. A careful up-sampling of those phrases with low occurrence or addition of new phrases has to carried out which will offer a better generalised model.

Deep learning has a lot of potential for sentiment analysis, as our results here proved that a basic CNN with only word vector representations as features perform quite well than other traditional classification techniques.

REFERENCES

[1] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115–124.

[2] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng and Chris Potts. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).

[3] Comparative Study of Machine Learning Approaches for Amazon Reviews Procedia Computer Science, ISSN: 1877-0509, Vol: 132, Page: 1552-1561, 2018

[4] C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library. (British Library Research and Development Report, no. 5587).

[5] Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Hafner. Gradient Based Learning Applied to Document. Proc. of the IEEE, November 1998.

APPENDIX

A. *model_CNN*

CNN Architecture Model

```
def model_CNN():
  model = Sequential()

  model.add(Conv1D(filters = 64,
      kernel_size=1, activation='relu',
      input_shape=(x_train_np.shape[1],1))
      )
  model.add(Conv1D(filters = 128,
      kernel_size=1))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(MaxPooling1D(pool_size=2))
  model.add(Dropout(0.25))
  model.add(Flatten())
  model.add(Dense(256))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dropout(0.5))
  model.add(Dense(5, activation='softmax
      '))
  return model
```

Listing 1. model_CNN custom method

B. *f1score*

F1 Score calculation

```
def f1score(y_true, y_pred):

  prec = precision(y_true, y_pred)
  rec = recall(y_true, y_pred)
  f1score = (2 * (prec * rec)/(rec + prec
      ))
  return f1score
```

Listing 2. f1score method

C. *test_model*

Load and Evaluate the Trained Model

```
from keras.models import load_model
def test_model(path):
  saved_model = load_model(path, compile=
      False)
  saved_model.compile(optimizer=adamax,
      loss='categorical_crossentropy',
      metrics=['acc', recall, precision,
      f1score])
  metrics = saved_model.evaluate(x =
      x_test_final, y = y_test_np)
  return metrics
```

Listing 3. test_model