Classifying Movie Reviews Utilizing Deep Learning Architectures

Lisa Ewen, 0655603
Department of Computer Science
Lakehead University
955 Oliver Road, P7B 5E1
lewen@lakeheadu.ca

***Abstract*— In this document it is noted that sentiment analysis is a useful NLP tool that can be applied to movie reviews in conjunction with, or in place of, the typical numerical rating scale. This was demonstrated previously by models utilizing SVM or Naive Bayes approaches, but little research has been seen regarding deep learning approaches. By using a one dimensional convolutional network utilizing four convolutional layers, four maxpooling layers, three dense layers and was trained on the Rotten Tomatoes training dataset, an accuracy of 64.4% was achieved. Despite not performing at the same level as other approaches, this model provides fantastic potential for future work in this area to discuss different deep network architectures that may yield even better results.**

I. Introduction

Text classification is an essential task in the field of NLP, featuring applications such as sentiment analysis, spam detection, and topic labelling. In particular, sentiment analysis is tasked with identifying and extracting meanings and opinions from documents with more subjective contexts such as posts from social media, forums, news websites, and reviews.

In this report we will focus on analysing and classifying the sentiment of movie reviews, which has a broad range of applications. As seen on many reviewing platforms, reviews tend to be ranked on a numerical scale with a lower score being a more negative or critical review. Users are also able to input text descriptions of why they gave the score they did, which is often more useful than the numerical score itself. Because of this, if websites utilized the sentiment of the reviews to determine the overall reception of a product or service, the ranking a prospective customer will see may have more impact and be more meaningful to that person. There are also possible applications for recommender services to utilize these more powerful ratings to make more accurate recommendations to customers.

We have seen this type of model demonstrated in some research efforts with sentiment analysis of movie reviews that will be discussed in Section II, but much of this research focuses on Naive Bayres or SVM based models with little deviation from those methods. This proves a marked lack of exploration with deep learning under this type of application, which in the following sections we will outline is significantly more powerful. While in this report we do not demonstrate any applications of the model that will be discussed in later sections for replacing a manual numeric scale of a review for reviewing platforms, there is obvious potential for said model to be used in this way. The model serves to provide a deep learning-based architecture for sentiment analysis of movie reviews that has not been discussed at length previously, and is meant to be a first step for future research for this topic.

II. Related Work

An experiment of sentiment analysis of movie reviews was conducted in 2006 using the Rotten Tomatoes dataset by Alyssa Liang [2], however, SVM and Naive Bayes models were used for classification. In all simulation results, relatively good accuracies were achieved; between 66% up to 87% when using a linear SVM in combination with TF-IDF and upweighting techniques. Since the work done by Liang, many advancements have been made in the field of NLP that were not fully developed at the time of that experiment. This was exemplified in a case study conducted by Sahu and Ahja in 2016 [4], where the IMDB movie review dataset was used and is pre-processed using Porter Stemming, Stopping, and POS Tagging. After pre-processing, the data is given a sentiment score based on features extracted from the review (positive and negative sentiment words, positive and negative sentiment bigrams, etc.), and classification techniques are applied. Such classification techniques include SVM, Random Forest, Naive Bayes, etc. Similar to the previously discussed model, experimental results yielded good accuracy with the highest result being an accuracy of 88.95% when using the Random Forest classification technique.

Similar approaches are seen in the work of Tsutsumi, Shimada, and Endo [6], Pang, Lee, and Vathyanathan [3], and Kennedy and Inkpen [1]. All of these models feature different pre-processing steps, architectures, however, all models seem to be primarily based on SVM or Naive Bayes structures. It is also important to note that many of these approaches are not recent, and did not have the capabilities of deep learning that are present in the modern field of NLP.

With the introduction of deep learning, NLP tasks have become easier and more accurate than they were previously [5]. Most notably, we have seen significant advances in NER, machine translation, POS tagging, and sentiment analysis that were not possible with previous model architectures. This suggests that under the umbrella of sentiment analysis of movie reviews, it is possible to utilize a deep learning architecture to provide similar or better results than have been achieved using structures like Naive Bayes or SVM models that we have discussed in this section.

III. Proposed Model

A. Data Preprocessing

Each phrase in the dataset first has stopwords and punctutations removed as these provide little importance for the context of the sentence in this case. Lemmatization is also used, increasing the time required for pre-processing, but providing with better accuracy during training. After these processes are complete, the data is split 70:30 using stratification and the training set is vectorized using TF-IDF. Stratification is necessary as the number of observations in each category are unevenly distributed which can be seen by table 1.

TABLE I: Sentiment Category Distribution

Sentiment Category	Number of Observations
0	7072
1	27273
2	79582
3	32927
4	9206

Additionally, due to the categorical nature of the data, the labels (called "Sentiments" in the dataset) are one-hot encoded on both the training and testing sets. This allows the data to be more expressive, and simplifies calculations as direct categorical data is often difficult to work with.

B. Considerations

To prevent overfitting, a dropout layer was included in the model with a value of 80%. In addition to dropout, a Maxpooling layer is added after each convolution operation which significantly reduced the overfitting problem.

In this case, smaller filters (3x3 and 5x5) are implemented to reduce the amount of computation required despite requiring an increase in the number of layers needed to properly detect each feature. The smaller kernel sizes also allow for more complex features to be detected, whereas larger filters may only detect features at a surface level.

C. Model Architecture

The model features four one-dimensional convolutional layers each followed by a maxpooling layer, one dropout layer, one flatten layer, and three dense layers whose architecture is shown in figure 1.

It is noted that the size of the convolutional layers gets reduce smaller after each layer to provide better performance. The first two convolution operations feature 100 5x5 filters, and the following two operations feature 120 3x3 filters. The same approach is taken with the dense layers where the number of neurons used in each layer are 100, 50, and 5 respectively. All layers utilize the ReLu activation function, aside from the final dense layer that has a Softmax activation function due to the multi-class categorical nature of the labels which have been one-hot encoded.

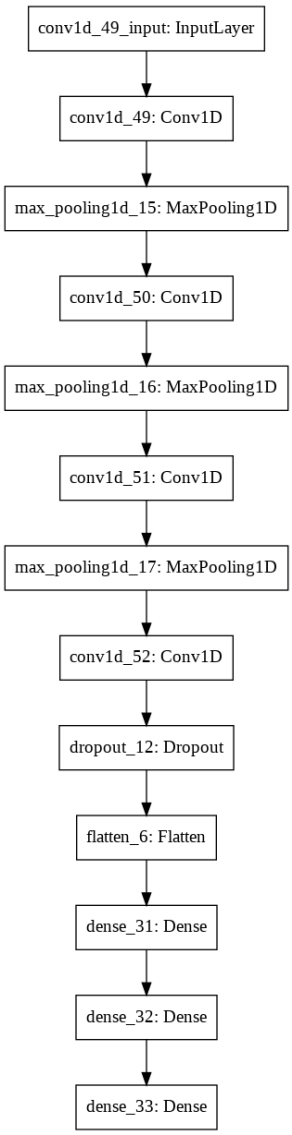

Fig. 1: Model Architecture

Evaluation Methods The model utilizes Categorical Cross Entropy loss and is evaluated on five scores: loss, recall, precision, accuracy, and F1 measure. Accuracy in this case is referring to the simple calculation of the number of correctly predicted outputs over the number of total outputs. To calculate recall, we utilize the following equation

$$\frac{\sum_{i=1}^{n} TP_n}{\sum_{i=1}^{n} TP_n + FN_n} \tag{1}$$

Where n is the number of predicted values, TP are the true positives, and FN are the false negatives. Similarly, precision is given by

$$\frac{\sum_{i=1}^{n} TP_n}{\sum_{i=1}^{n} TP_n + FP_n} \tag{2}$$

Where n is the number of predicted values, TP are the true positives, and FP are the false positives. The F1 Measure is simply represented by the equation

$$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

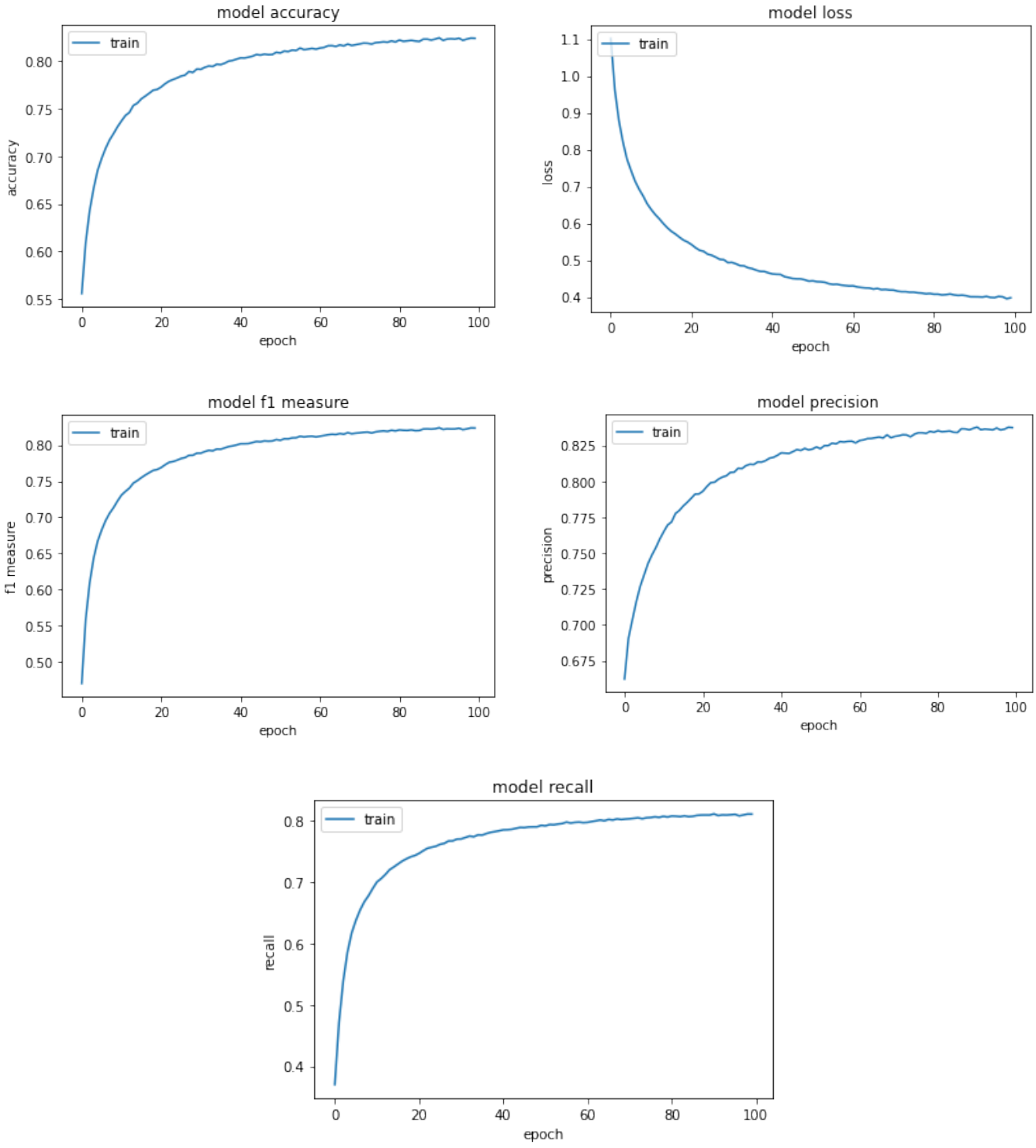

Fig. 2: The plots for accuracy, loss, F1 Measure precision, and recall per epoch

These three equations were implemented using the functions shown in listings 1, 2, and 3.

Listing 1: Recall

```
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
```

Listing 2: Precision

```
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
```

Listing 3: F1 Measure

```
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()))
```

IV. Experimental Analysis

Simulation tests of the model were conducted over 5 separate runs of the model which was trained over 100 epochs, achieving an average accuracy of 64.4%, precision of 66.8%, recall of 60.4%, F1 Measure of 63.4%, and training time of 183.4 minutes. Figures 2 displays the training progress of the the first run:

The individual results of each run are displayed in the following table:

TABLE II: Simulation Results

Run #	Training Time	Accuracy	Loss	Precision	Recall	F1 Measure
1	183	0.64	0.95	0.66	0.60	0.63
2	189	0.65	0.93	0.67	0.61	0.64
3	182	0.64	0.94	0.67	0.60	0.63
4	180	0.65	0.92	0.67	0.61	0.64
5	183	0.64	0.93	0.67	0.60	0.63
Average	183.4	0.644	0.934	0.668	0.604	0.634

Given these results, we can extrapolate that the overall performance is decent all scores. Most notably, the models discussed in Section II performed at slightly higher rate, but the performance is close to the proposed model with a more complicated network structure. This is significant due to the relatively primitive structure utilized in the proposed model in comparison to the previously discussed approaches. With a more complicated and deeper network implementation in combination with other processing techniques (such as Word2Vec) and larger amounts of training data, we could expect to see performance that exceeds the previously discussed models due to the powerful deep learning architecture.

V. Conclusion

In many ways, sentiment analysis has provided plenty of beneficial applications. That being said, there are still ways to utilize sentiment analysis to improve systems, such as ranking systems, in ways that have not been fully explored. We have previously seen sentiment analysis of movie reviews done utilizing Naive Bayes or SVM architectures, but little research has been done in this area utilizing deep networks. In this report we have seen that a model can be constructed to determine the overall opinion a movie review within 64.4% accuracy while utilizing a one dimensional convolutional network. This model implementation serves to prove that a simple model empowered with deep learning can perform at almost a similar rate as more complex models using more primitive structures. This provides opportunities for future exploration with different pre-processing techniques and a deeper model representation to provide unprecedented results for this type of application. We may also see benefit from training this model on different sets of movie reviews from different sources which may give a better idea of how generalizable this type of model is, and provide future insights for how a model like this could be best utilized among different applications.

References

[1] Alistair Kennedy and Diana Inkpen. "Sentiment classification of movie reviews using contextual valence shifters". In: *Computational intelligence* 22.2 (2006), pp. 110–125.

[2] Alyssa Liang. "Rotten tomatoes: Sentiment classification in movie reviews". In: *CS* 229 (2006), p. 15.

[3] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques". In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics. 2002, pp. 79–86.

[4] T. P. Sahu and S. Ahuja. "Sentiment analysis of movie reviews: A study on feature selection classification algorithms". In: *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*. Jan. 2016, pp. 1–6. DOI: 10.1109/MicroCom.2016.7522583.

[5] Tryolabs. *Deep Learning for NLP: Advancements Trends*. Dec. 2017. URL: https://tryolabs.com/blog/2017/12/12/deep-learning-for-nlp-advancements-and-trends-in-2017/.

[6] Kimitaka Tsutsumi, Kazutaka Shimada, and Tsutomu Endo. “Movie review classification based on a multiple classifier”. In: *Proceedings of the 21st pacific Asia conference on language, information and computation*. 2007, pp. 481–488.