Sentiment Analysis of Movie Reviews with Neural Networks: A CSC 439 NLP Project

Tugay Bilgis
University of Arizona, Tucson, AZ, USA.
tbilgis@email.arizona.edu

Abstract

Movie reviews are an important factor on deciding the quality of a movie and there is a lack of resource on automatically determining the score of a movie with the written reviews. In this paper, we focus on automatically labeling the movie reviews as either positive or negative. We create two models, one baseline and one more complex model using Neural Networks that performs better than the baseline model we created. We show that we can achieve accuracy around 87% and show how much better it performs against the baseline model.

1 Introduction

Sentiment analysis is the process of automatically identifying whether a piece of text is conveying a positive, neutral or a negative idea. With the development of the internet, this task has become a hot topic in the Natural Language Processing area.

Sentiment analysis has broad utility, to give an example, it can help us determine the general consensus about a hot topic that is on social media by analyzing each post about that topic. Automatically analyzing these posts can influence the act of politicians and help them raise their vote count.

Movie reviews is a popular area for sentiment analysis. By doing a sentiment analysis on movie reviews posted by users of web forums or in sites like IMDB, we can determine the general rating from the public and that rating could give us a general public rating on a movie rather than the critics' ratings. The sentiment analysis for movie reviews have been studied widely, but in this work, we strive to create a model that can outperform the state-of-art models and explore how Neural Networks could be a potential candidate for it.

Data with pre-labeled reviews is widely available. In this work, we used the dataset that was prepared by a group at Stanford University (Maas et al., 2011) and contains 50,000 movie reviews that was posted on the IMDB website.

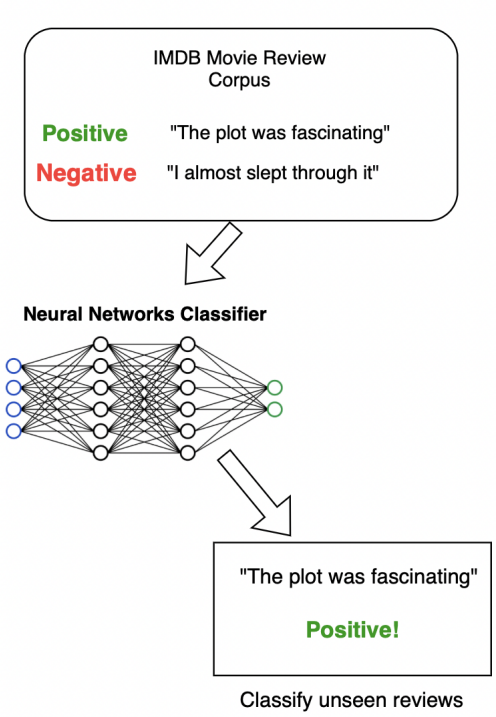

Figure 1: An example of the sentiment analysis task. A corpus of IMDB movie reviews serves as training data to fine-tune a Neural Networks classification model, trained to classify reviews as *positive* or *negative*. The classifier is then evaluated on unseen reviews.

We first create a Random Forest classifier to base our experiment with the model that we are going to create. Then we create a neural network model that is designed to do binary classification. We split our data into three: train, validation and test. We train the model with the train data, validate it with the validation data, and as seen in Figure 1, we test our model by inputting the test data that our model hasn't seen yet and get results of around 87% accuracy.

In this paper, we discuss how we conduct our experiment, and the results that we have gotten. We compare our results with the baseline model to show the difference between our model. We analyze the results and conduct an error analysis to detect where our model fails and what can be done

for future improvements.

2 Related Work

There has been a lot of previous work done by other researchers in the sentiment analysis area. The previous works focuses on variety of different areas in sentiment analysis. We have found three papers that aligns with our work and represents the work done in the past.

Pang et al. (Pang et al., 2002) provides the first detailed study on the sentiment analysis of movie reviews. They identify the problem as a binary classification algorithm. They use a dataset from IMDB, but with a much smaller sized one compared to ours. They implement n-gram classifiers such as Naive Bayes and SVM to perform the sentiment analysis. They get a result of 80.6% accuracy with a Naive Bayes classifier with bigram and unigram features. They reported that SVMs performed the best amongst the n-gram classifiers. Compared to Pang et al., our work includes a much larger training evaluation, and uses a Neural Networks model that improves the results of n-gram classifier models such as Naive Bayes and SVM.

Batanovic et al.(Kapukaranov and Nakov, 2015) provides a study on the sentiment analysis of Bulgarian movie reviews. They identify the problem as multi-classification problem and therefore offer a fine-grained sentiment analysis on the Bulgarian movie reviews. They use SVM to do a multi-class classification problem. Their textual features only contains bigrams, but they have added other contextual features such as country, genres, actors, etc. They use a different metric than us, but our model has a larger dataset due to our sentiment analysis in English.

There have also been other researches conducted on different areas of sentiment analysis. Yang et al. (Yang et al., 2020) has worked on the sentiment analysis of customer reviews that has been left on the websites. Their approach differs as they combine the summary and the review, and create a more encoding-based model. Their results suggest that they have been able to increase their model accuracy when they used a joint encoder that includes the summary and the review. Our model does not implement joint encoding, but it could be a future implementation to our model.

Overall, there has been a good amount of research that has been done in sentiment analysis and also specifically in movie reviews. We make use of the past research and develop a new solution to the problem with a model that has not been extensively studied and one that could potentially be the state-of-art for movie review sentiment analysis.

3 Approach

Movie review sentiment analysis can help us determine the overall rating for a movie, by doing sentiment analysis on each review. Our problem was to see if our model can determine an unseen movie review as a positive or a negative review. For this reason we decided to frame it as a classification problem. It is possible to frame it as a multi-class classification problem to analyze the reviews in-depth and get a fine-grained analysis and rating, but for this instance, our approach was to do a binary classification since it would be sufficient for this task.

... it was well-paced, spectacular job. → positive
... the plot was really boring. → negative

Given an unseen movie review, we build a model that will predict the likelihood of the review being positive or negative.

3.1 Data collection

We used a dataset that already existed. The dataset that we used for this model was retrieved from a group of researchers at Stanford University (Maas et al., 2011). The dataset provides 25,000 highly polar movie reviews and 25,000 for testing. Each of the movie reviews come from the IMDB website. Each review is labeled with either 1 or 0 for a binary sentiment analysis, with 1 indicating a positive review and 0 indicating a negative review.

Positive: 1
"Very smart, sometimes shocking, I just love it. It shoved one more side of David's brilliant talent. He impressed me greatly! David is the best. The movie captivates your attention for every second."
Negative: 0
"Cool idea... botched writing, botched directing, botched editing, botched acting. Sorta makes me wish I could play God and strike everyone involved in making this film with several bolts of lightning."

There is an even number of positive and negative reviews, therefore randomly guessing gives a 50% accuracy. There are at most 30 reviews for each movie to limit the number of reviews per movie.

We split the dataset as followed, 50% for training, 25% for validation and 25% for testing. We used the already existing 25,000 reviews on the training set provided to train our model and split the testing set in two for validation and testing.

3.2 Experiments

We created two models that are capable to do binary classification. The first model is the baseline model, and the second model is the model that we tried to achieve a better performance and benchmark.

To evaluate the performance of these two models, we will compare both model's F1 and accuracy score.

3.3 Models

Our baseline model is a simple n-gram model that uses a common classification algorithm and one that is widely used. This model is used as a benchmark to evaluate how better is the complex model that we created.

Our complex model is a neural networks model that performs a binary classification task.

Baseline Model: For our baseline model, we implemented a simple Random Forest classifier, which tends to work well with text classification. The *scikit-learn* framework is used to implement the features, encodings and the classifier.
We used the following textual features:

- **words**: binary feature for each word,
- **n-grams**: binary feature for each n-gram. We only used unigrams for the baseline model.

Baseline Model	
Accuracy	82.4%
F1 Score	82.5%

Table 1: Task performance across the baseline model under investigation.

Recurrent Neural Networks Model: For our model, we created a recurrent neural network, which makes use of the long short-term memory network. Specifically, our model takes information from prior inputs to influence the current input and output. We made use of Tensorflow, specifically the Keras API. We stacked two long short memory layers to make our model deeper. The LSTM hidden size is set to 64. Our model performs 10 epochs. We use Adam to optimize the model, with a learning rate of $1e$-4.

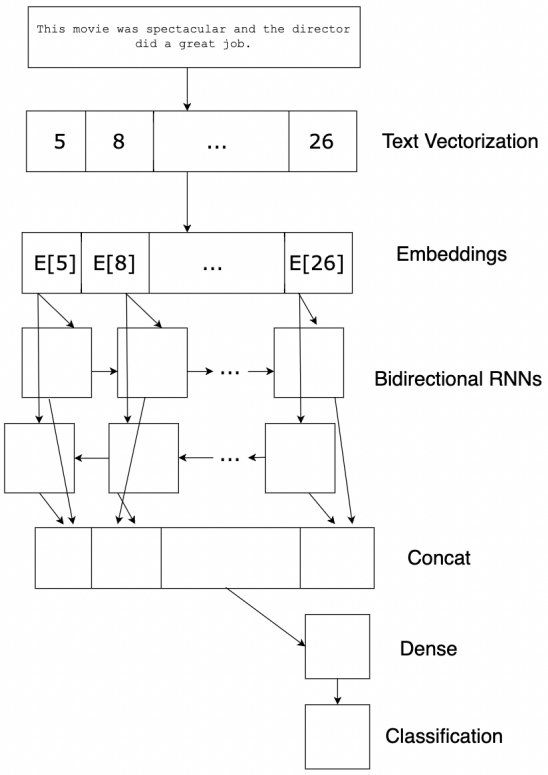

Figure 2: Visual representation of our model

We make use of TextVectorization from the Tensorflow for our features, which is a layer that converts the given words into integer sequences to be used with the other layers. Unigrams and bigrams are used in this model for text features.

RNN Model	
Accuracy	87.61%†
F1 Score	87.66%

Table 2: Task performance across the RNN model under investigation. † signifies that performance is significantly different ($p < 0.05$) using a non-parametric bootstrap resampling.

3.4 Hyperparameter Tuning

We first implemented the RNN model with only one long short-term memory layer and with only unigram features, and we got an accuracy score of around 86.4%. We then decided to add an extra long short-term memory layer and use both bigram

and unigram features on the development set to see if it increases the accuracy on the test set, and we were able to increase our accuracy up to 87.61%.

3.5 Results

The experiment results are described in Table 1 and 2. Overall the results show that our RNN model performs significantly better than the baseline model. Non-parametric bootstrap resampling was used to determine how significant the difference is. We got a p-value of 0.03 with 10000 resamples.

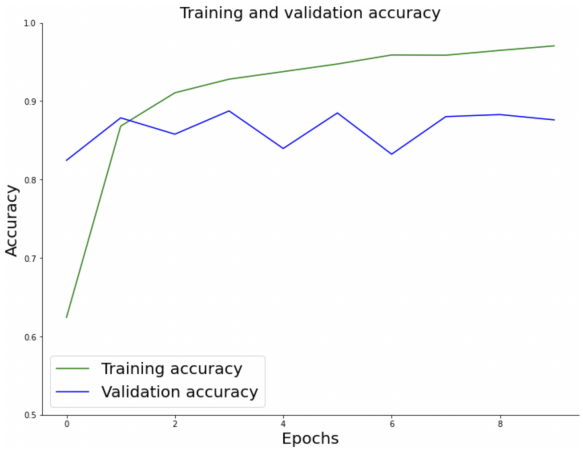

Figure 3: Learning curve of our model

The learning curve of our model indicated that our model did not overfit or underfit and that it is likely to be a good fit.

3.6 Error Analysis

We analyzed randomly selected 50 samples from the reviews that were labeled incorrectly by our model. Those are shown in Table 3 and described below.

Unique Names - Low frequency words (90%): Majority of the errors in our model was from the reviews that contained the names of the characters from the movie, the movie's name or the actor/director's name, which it was not trained on. An example is shown below:

*"I did not think **Haggard** was the funniest movie of all time I like **CKY** and **Viva La Bam** a lot more ... **viva la bam, CKY, or Jack Ass** should see it ... I think that **Jonny Knoxville** should ... from **Haggard** and ..."*

Sarcasm (6%): Around 6% of the errors that we have seen included sarcastic reviews. The sarcastic reviews uses a lot of the words that appear in the opposite reviews, therefore causes our model to predict them incorrectly.

Complex reviews with ambiguous words(4%): Some movie reviews that were detailed and contained ambiguous words caused our model to make wrong predictions.

Prop.	Error Class
90%	Reviews with unique names and words
6%	Complex reviews with ambiguous words
4%	Reviews that contain sarcasm

Table 3: Common error classes and proportions of errors for 50 randomly selected errors on the development set.

4 Conclusion

We created a binary classification model that is designed to do sentiment analysis on the movie reviews from the IMDB. Our model was created using Recurrent Neural Networks. We used a baseline model of a Random Forest classifier. Our RNN model performed vastly better than the baseline model that we have implemented, giving us an accuracy score of 87.61% and an F1 score of 87.66%. Tuning up the layers of the neural network gave us better results. The learning curve for our model indicates that our RNN model is a good fit for this problem. The error analysis shows that our model is mostly doing wrong predictions with the reviews that contains a lot of unique words, such as actor names and character names. Our work is comparably better than the previous work done and provides a new perspective on the sentiment analysis of movie reviews. Our future work will be on training on more data with unique words and explore other classifiers to perform a better movie sentiment analysis.

5 Project Site

The project is available at https://github.com/tbilgis23/CSC-439-Final-Project .

References

Borislav Kapukaranov and Preslav Nakov. 2015. Fine-grained sentiment analysis for movie reviews in Bulgarian. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 266–274, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, page 79–86, USA. Association for Computational Linguistics.

Sen Yang, Leyang Cui, Jun Xie, and Yue Zhang. 2020. Making the best use of review summary for sentiment analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 173–184, Barcelona, Spain (Online). International Committee on Computational Linguistics.