Amanda Li, Pearl Vishen, and Hannah Wen

ECS 111: Applied Machine Learning

Professor Gabriel Simmons

March 15, 2024

A Neural Network Approach to Analyzing Letterboxd Movie Reviews

Introduction

In the realm of sentiment analysis, understanding the expressions of younger individuals presents a unique challenge. This project explores the humorous and often irreverent writing style found in reviews on Letterboxd, a popular platform for film enthusiasts. Given the absence of a comprehensive dataset of Letterboxd reviews, we scraped reviews directly from the website. Our focus lies in predicting sentiment polarity, using the existing ratings as ground truth. We preprocess the data for training a shallow neural network model. Our methodology involves fine-tuning the network's architecture and parameters to optimize performance. Our final model includes an accuracy score of 0.50 and an F1 score of 0.51. This research not only contributes to the field of natural language processing but also highlights the evolving communication styles of the digital generation.

The dataset utilized in this study was acquired through web scraping in Beautiful Soup on Letterboxd data from 2023. We focused on the initial 500 pages featuring the top-rated movies of all time, resulting in 3,600 movies represented in our dataset. Initially, we collected the reviews and associated movie data, but numerical ratings were not included. However, we successfully matched these ratings to our pre-collected movie reviews to use as ground truth data. Our original dataset consisted of 60,000 observations, which we eventually whittled down to 32,551 representing 1967 movies after data cleaning. The review ratings are on a scale from 1 to 10, representing a 5-star rating system, including half-stars. Each point on the scale corresponds to half a star. The reviews were mostly positively reviewed, meaning the data did not have an equal distribution of negative and positive reviews. To handle this imbalanced data, we used class weights and resampling techniques.

Our dataset was cleaned of all emojis and punctuation, and all the words were turned into lowercase. Contractions such as were expanded to with the contractions package. For example, “it’s” would be transformed into “it is”. We created 2 datasets: one without the stopwords defined in the NLTK package and one without any words removed. We ended up using the dataset with stopwords to evaluate our model because we noticed that many stop words were important to the reviews, especially since so many were fairly short. We created 3 sentiment classes: positive, neutral, and negative. The reviews on Letterboxd were given a rating scored from half a star to 5 stars, translating to a rating of 1-10 with 1 star represented by 2 units. We defined positive as reviews rated 8-10 (4 to 5 stars), neutral as reviews rated 5-7 (2.5 to 3.5 stars), and negative as reviews rated as 1-4 (half star to 2 stars). We then one hot encoded these labels. The negative class is represented by 0, the neutral class is represented by 1, and the positive class is represented by 2.

While exploring the data set, it was observed that the ratings were imbalanced. As a result, the classes will also be deeply imbalanced. Negative reviews are vastly underrepresented compared to positive reviews. There are more neutral reviews than negative reviews, but are still underrepresented compared to positive reviews.

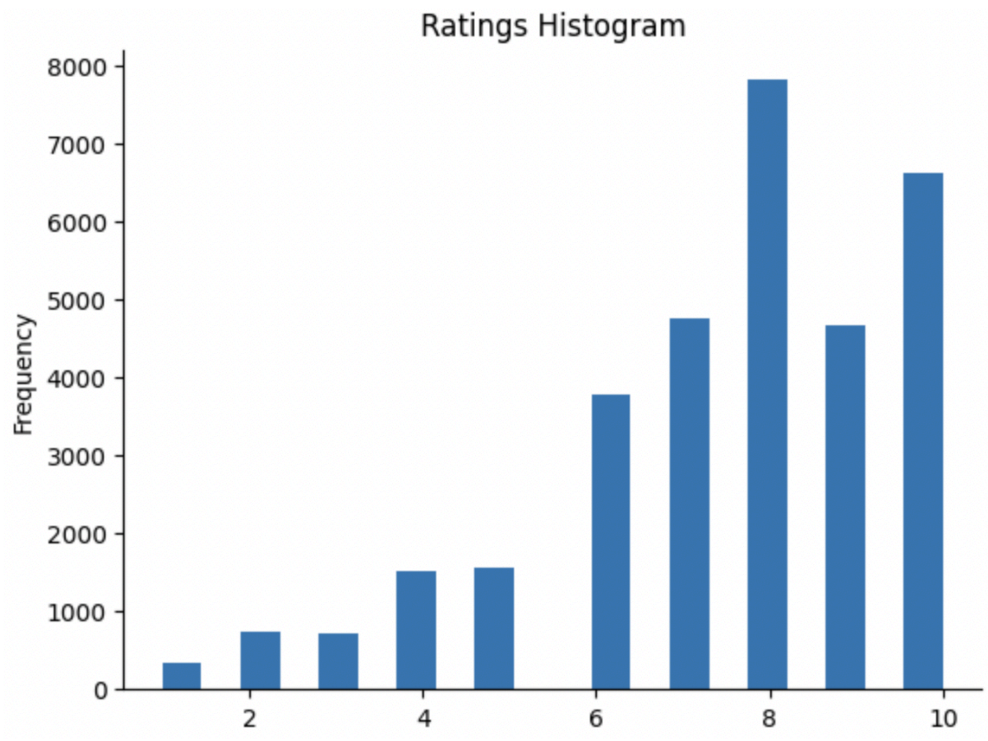

Figure 1: Histogram of the ratings among the web scraped reviews.

We used a recurrent neural network (RNN) architecture for sentiment analysis due to its ability to understand each word based on the understanding of previous words (Olah, 2015). Moreover, we added a Long Short-Term Memory (LSTM) layer to mitigate the vanishing gradient problem that occurs in RNNs (Hochreiter, 1998; Nowak, 2017). After settling on the topic of a neural network, we approached this project with several ideas in mind.

Simple RNN Method (Trial 1)

One initial method was to build a neural network to predict the rating assigned by the user, which would be a number scaled from 1-10 representing the range of a half star to 5 stars. We first tried this idea with categorizing the ratings as classes. After getting a training accuracy of 19%, we realized that we had too many classes and that accuracy would not be the right metric to evaluate our rating predictions because a rating of eight misclassified as a rating of nine would be considered inaccurate, despite the prediction being very close. While we could have switched to predicting a numerical rating, rounding it, then measuring model performance with metrics such as MSE, we decided to generalize the ratings to positive, neutral, and negative for simplicity. We also did this because of the class imbalance within the positive, neutral and negative groupings. For example, there were many more four star ratings than five star ratings.

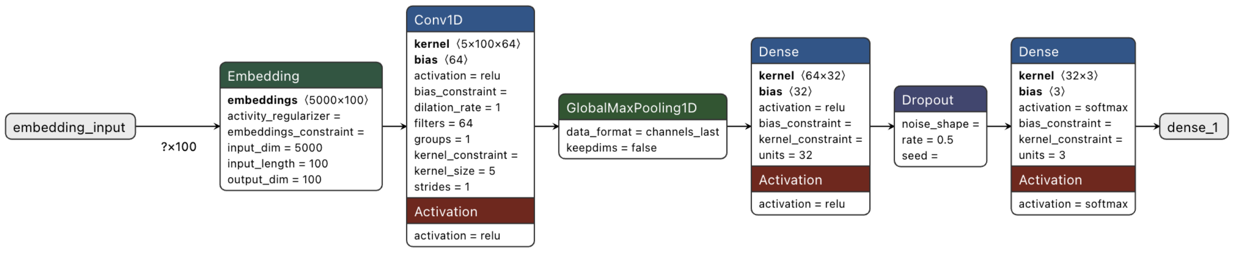

Figure 2: Boilerplate RNN with five layers (Kuria, 2023).

Transformers Method (Trial 2)

Moving forward, we attempted pre-trained word embeddings and algorithms. We first experimented with BERT (Bidirectional Encoder Representations from Transformers), due to the Transformer's ability to use attention mechanisms to compute representations of input and output sequences (Vaswani et al., 2017). However, the BERT tokenizer combined with our dataset was computationally expensive. Our memory RAM was overloaded and our session would often crash. Even using the smaller scaled DistilBERT still was too computationally expensive for us. We tried Word2Vec as an alternative by trying both training word embeddings and using pre-trained word embeddings. When attempting to train our own word embeddings on the pre-trained algorithm, we ran into an error regarding vocabulary size that kept occurring despite multiple alterations and cross-checks. When trying the pre-trained word embeddings we also had errors with vocabulary size; we could not implement an embedding layer because the vocabulary size could not be found. Our model was a very simple LSTM model. While running the neural network, the training accuracy ranged from 29-34%, and the validation accuracy would often end up very low: around 9-10%. Loss did not decrease much with each epoch during training even after alterations were made. Essentially, it appeared that the model was not correctly training.

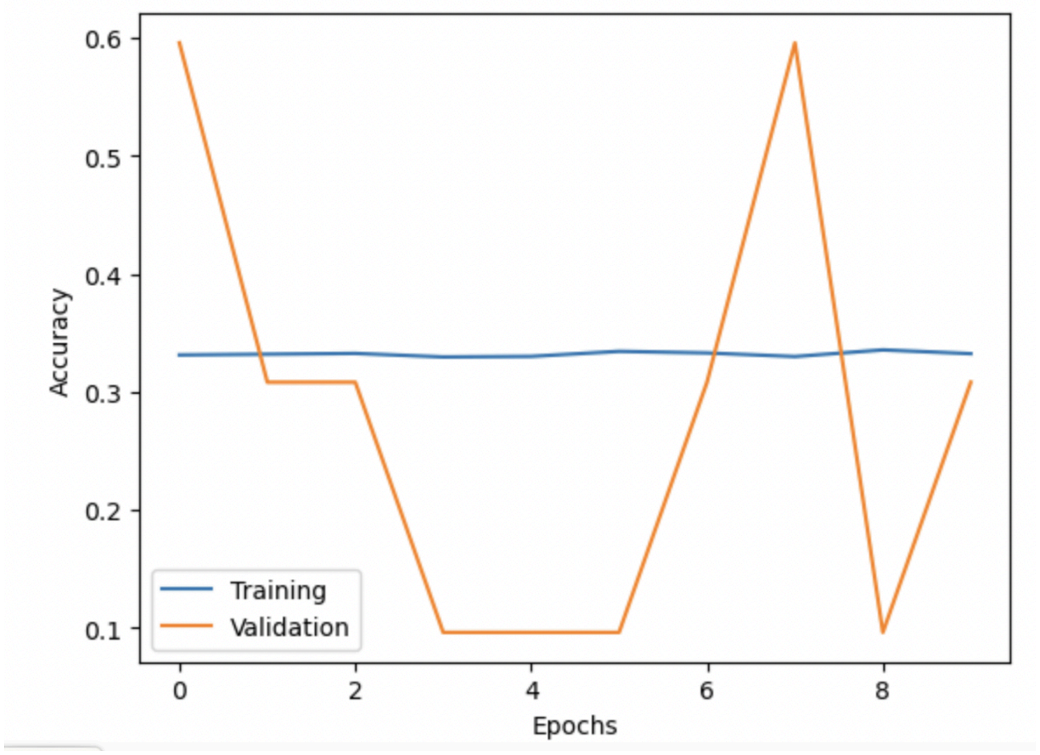

Figure 3: Training and Validation Accuracy over 10 Epochs for the Word2Vec Model.

Generally, the vocabulary size threw many errors, and we ultimately decided to try creating word embeddings without additional models or data.

Results: Final Model with LSTM (Trial 3)

After attempting to create a more complex neural network with pre-trained models and embeddings, we decided to go back to our original boilerplate model and expand from it. We utilized the TensorFlow tokenizer and fit it on our training data, with a vocabulary size of 372,160 words. The reviews were truncated and padded to 100 words before being turned into sequences. We changed the train/test/validation data set proportions to 70/15/15 instead of 80/10/10 to better evaluate the model and have the model adjust its hyperparameters according to the validation set performance. To deal with class imbalance, both class weights and SMOTE (Synthetic Minority Over-sampling Technique) were implemented on different models. The model with class weights performed slightly better with an accuracy and weighted F1 score of

53.2% and 0.527 compared to the accuracy and weighted F1 scores of 48.9% and 0.499 for the model which had its training data oversampled with SMOTE. The SMOTE dataset was roughly double the size of the original X train dataset with 40026 observations compared to the 22785 observations in X train, resulting in a more computationally expensive runtime. The precision scores among classes for each model varied very slightly depending on each class, while the recall and F1 scores among classes were more even for the SMOTE model. Due to the small differences between recall and precision metrics, the larger weighted accuracy by approximately 3%, and after considering our computational limits, the model with class weights was chosen to be finalized into our final model. This model consisted of an embedding layer, a ReLu activation convolutional 1D layer with 64 filters and a window size of 5, a global max pooling 1D layer, a ReLu activation dense layer with 32 neurons, a dropout layer set to 0.5, and a soft max dense layer with 3 neurons. The convolutional 1D layer was later on replaced with a LSTM layer with 128 filters and a dropout rate of 0.2, resulting in our final model. The overall testing accuracy was lower by approximately 1% but we continued with this model because it overfit less than the previous one.

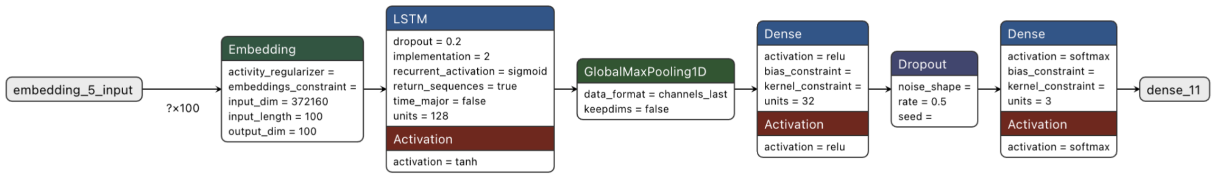

Figure 4: Original RNN, but with Conv1D replaced with an LSTM layer and class weights added.

Our final model had a training accuracy of 87.25% and a validation accuracy of 49.31%, which still indicates overfitting. The precision scores for the negative, neutral, and positive classes are 0.2, 0.37, and 0.67, respectively. The recall scores for the negative, neutral, and positive classes are 0.33, 0.37, and 0.6, respectively. The F1 scores for the negative, neutral and positive classes are 0.25, 0.37, and 0.63, respectively. These uneven scores still reflect the imbalanced classes in the data, despite the use of class weights. The testing accuracy was 50.4% and the weighted F1 Another model was made with a batch normalization layer to prevent overfitting, but precision and recall scores became more imbalanced among classes, despite a slightly higher accuracy and weighted F1 score of 51.1% and 0.521, respectively. All models were run with 10 epochs and a batch size set to 64. They were all compiled with the Adam optimizer, accuracy as the model metric, and categorical cross entropy as the loss function.

Discussion and Limitations

There are several reviews that have little to do with the numerical rating given by the user. For example, there is a review for the movie *Megamind vs. the Doom Syndicate (2024)* that states “Happy to see American animation is alive and well!” with a 0.5 star rating corresponding to it. This review would probably be classified with positive sentiment despite the negative rating. When this review was fed into our final model, it was classified as positive. Initially, we viewed these reviews as noise in our dataset. However, we came to realize that there was a lot more of this noise than we thought. Scrolling through our dataset, we observed that many of the reviews did not indicate the user’s rating of the movie. In fact, even if a human were to manually predict the sentiment of the review, the accuracy rate may not be high. Additionally, the language in the reviews is often contextual referencing other pop culture. Sarcasm and other types of humor are very common throughout user reviews in Letterboxd, which made assigning a sentiment more challenging.

We employed resampling techniques to ensure that our model was robust. However, after resampling, we observed that the overall accuracy reduced by about 0.33. In the future, this could be avoided by utilizing a balanced dataset, with an equal amount of reviews in each class. Moreover, it would be beneficial to use a dataset where each review contained a rating, rather than having some reviews with missing ratings that add to the noise.

Conclusion

The addition of an LSTM layer within our model proved to be the most robust choice, which mitigated issues such as the vanishing gradient problem commonly associated with RNNs. By using the LSTM's ability to sustain information over longer sequences, we aimed to capture the subtle cues that are present in the informal language of Letterboxd reviews. Our final model demonstrated significant improvements in both accuracy and performance metrics compared to earlier iterations. Our decision to use a model with LSTM has also been backed by recent literature, where LSTM proved to be the superior strategy compared to ANNs and CNNs in an analysis of IMDB reviews (Hossain et al., 2022). With a training accuracy of 87.25% and a validation accuracy of 49.31%, our model shows reasonable capability to classify sentiment polarity within the dataset. However, these metrics were derived from overfitting, which is something that should be optimized in future iterations.

Contribution

Amanda - I defined the boilerplate RNN for our model, I contributed to EDA by creating BERT topic modeling visualizations, and I wrote the bulk of the report.

Hannah - I collected and cleaned the reviews, worked on the exploratory data analysis and testing various models, and finalized our model and evaluated its performance.

Pearl - I researched pre-trained word embeddings and worked with Word2Vec to develop a model. I cleaned and organized code files for submission.

References

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *06*(02), 107–116. https://doi.org/10.1142/s0218488598000094

Hossain, S. Md. M., Sumon, J. A., Alam, Md. I., Kamal, K. Md. A., Sen, A., & Sarker, I. H. (2022). *Classifying Sentiments from Movie Reviews Using Deep Neural Networks*. Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-031-19958-5_37

Kuria, D. (2023, March 26). How to create a sentiment analysis model from scratch. *MakeUseOf*. https://www.makeuseof.com/create-sentiment-analysis-model/

Nowak, J., Taspinar, A., & Scherer, R. (2017, January 1). *LSTM recurrent neural networks for short text and sentiment classification*. Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-319-59060-8_50

Olah, C. (2015, August). *Understanding LSTM Networks -- colah's blog*. Github. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–6010.