Product Review Sentiment Analysis

Kevin S. Laudano
ksl222@lehigh.edu
Data Cleaning, Model Evaluations

Jiageng Zheng
jiz322@lehigh.edu
Deep Learning Models

1 Abstract

This project uses sentiment analysis techniques to predict the number of stars assigned to an Amazon product review based on the review's text. Sentiment analysis involves figuring out the positivity/negativity or emotions in a text. To accomplish this, a recurrent neural network (RNN) and several other deep learning models were trained to predict whether a given review was classified as 1, 2, 3, 4, or 5 stars.

2 Introduction

Product reviews are important because they provide feedback that informs a company of what they are doing well and what needs improvement. In addition, reviews help customers decide what products to purchase. Thus, being able to understand whether people like or dislike a product is vital.

There are a number of challenges involved with trying to understand how much a customer likes a product based on what they write in a review though. There are, of course, the usual difficulties in NLP projects such as misspelled words and improper syntax which can be reduced by properly cleaning up the text data. In addition, reviews may have conflicting views on a product. For example, a customer may like one aspect of the purchased item, but may be dissatisfied overall. One model previously used for this task, Naïve Bayes' Classifier, can overcome this problem by taking into account the number of words generally found in positive reviews in comparison to the number of words generally found in negative reviews. The strength of the language used must also be considered (e.g. "great" is stronger than "good" and more likely to indicate a positive review). Again, Naïve Bayes' Classifiers, for example, would account for this because stronger language is more likely to appear in polarized reviews (i.e. reviews that have very few or many stars) than in more moderate reviews.

In order to accomplish this task, the first thing that needed to be done was cleaning the data. The raw data contained extra information (e.g. product ids) which needed to be removed, so only the review text and review ratings remained. Each of the review texts also needed to be cleaned up to make training the machine learning model easier. In order to improve model performance, the number of reviews of each class were balanced. (Some of the reviews from the classes that made up a larger portion of the dataset were removed.) After that, multiple models were trained on the review text to predict the likelihood of the review belonging to each of the classes; 1, 2, 3, 4, and 5 stars. The deep learning models used in this project include several different RNN architectures, and a tuned, pretrained Bidirectional Encoding Representation from Transformers (BERT) model.

Based on our study, the RNN outperforms many other models, such as naive bayes classifiers and artificial neural networks (ANN); it has higher accuracy, precision, recall, and f1 scores along with lower average absolute error. However, a fine-tuned BERT model tends to outperform RNN when trained and evaluated on balanced data sets.

3 Related Work

3.1 Seed Paper

The seed paper chosen looks at an interesting example of product reviews, "what happens when reviews do not match their ratings?" This mismatch between reviews and ratings can result in average product reviews that do not correctly represent how people feel about the product. So, the goal of the seed paper was to find a way of automatically identifying such reviews using deep learning.

Previous solutions to this problem included using bag-of-words and bag-of-ngrams to train naïve bayes classifiers and support vector machines (SVM). Although such techniques have performed adequately, there is room for improvement as they have drawbacks. Bag-of-words disregards the context of words and bag-of-ngrams can have high dimensionality.

In order to solve the problem, the paper used a gated recurrent neural network (GRU) to learn vector representations of reviews through paragraph reviews and product embeddings. The output of the RNN was then used to train a SVM which determined whether there was a mismatch between a review and its given rating.

The data used to train and test the model came from Amazon product reviews.

The model was evaluated by comparing the output of the model (whether there was a mismatch or not) to reviews in test data. The accuracy, recall, and precision of the model was calculated over the iterations of training. The best statistics achieved were an accuracy of 81.82%, recall of 45.52%, and precision of 59.45%. Although the accuracy of their model is not bad, it is clear that more work must be done before the problem can be considered solved.

Despite the fairly high accuracy of the model in the paper, it has poor recall and precision.

Further, the paper tries to ensure that ratings are representative of the product. However, it fails to take into account user information. Some reviewers may be prone to giving particularly low/high ratings which will affect the average product rating.

3.2 Other Papers

To complete this task, we read additional papers that performed the task with a variety of methods. Several works trained deep learning models to perform sentiment analysis on GloVe embeddings, unlike the seed paper which used paragraph2vec embeddings. Another paper trained an embedding layer to recognize meaningful word vectors. One paper we read mentioned several other embedding types that have been used for NLP, including Continuous Bag-of-Words (CBOW) and Skip-Gram (SG) models. While the seed paper used a gated recurrent unit (GRU), some of the other papers we studied used long short-term memory (LSTM) models. In terms of implementations, many papers used the Python library pytorch, however, keras was also a popular library for implementing the deep learning models.

4 Methodology

4.1 Neural Network Architecture

The main deep learning model we created for the task was a recurrent neural network, but an artificial neural network (ANN) and BERT model were also used.

4.1 Non-Recurrent Models

As baselines, models in this section measure the Amazon review using the sum of word embeddings. They do not capture any relationship between words. For example, “not good” will be interpreted as the sum of the words “not” and “good.”

4.1.1 Length of Five Embeddings

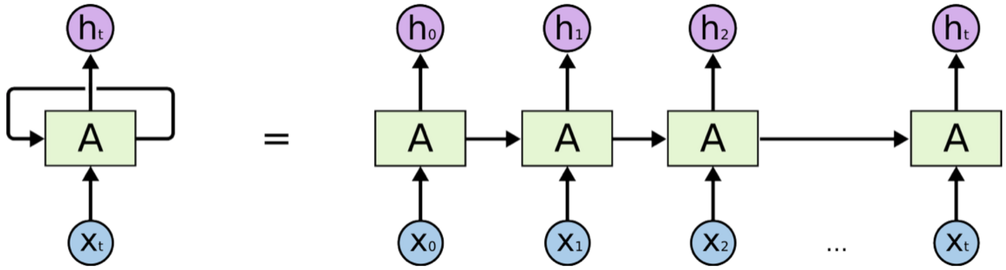
This is the simplest model. To evaluate the rate of a review, each word is embedded using a trainable vector of length five. Then, we compute the sum of each word embedding in a review, which results in a vector also having length of five. Passing through the softmax layer, the prediction of possibility is provided. The cross entropy loss is used to train this model.

4.1.2 Regression with ANN Classifier

In this model, we explored the viability of retrieving a scalar value for each word to represent its positive or negative attitude. A One Hot embedding is used, and a weight value for each word is trained. The l2 loss is used in this regression task. However, it does not give a classification of the rating, so we trained an ANN classifier to make this method comparable with others in this section. The cross entropy loss is used.

4.2 NN Architectures Using LSTM Layer

Models in this section contain one LSTM layer so they are able to model the relationship between words. The cross entropy loss is used for all training.



4.2.1 Weighted Sum of LSTM Output

Similar to the way of predicting POS tags, the output of the LSTM layer is used in this model. It gives a vector of output size for each word in a review. We set the output size to five, so we are able to replicate the same methods mentioned in 4.1.1. The accuracy seemed to increase after we introduced a trainable weight vector (similar to one in 4.1.2) to compute the weighted sum of LSTM outputs.

4.2.2 LSTM Last Hidden State

For all the review embeddings passed to the LSTM layer, we only retrieve the last hidden state (depicted as h_t in the figure) because this hidden state is able to provide an aggregated representation. Then, by adding a linear layer, we convert the vector of hid_dim to a vector of length five, passing through the softmax layer which generates the output of this model.

4.3 BERT

We tuned the pre-trained BERT model. Our previous models considered only 20,000 tokens over 340,000, but this BERT, by generating word embeddings in a delicately designed way, takes care of all 340,000 tokens. It is also memory intensive. To increase the speed of training, the batch size was lowered to 8. Each epoch takes 2 hours and tends to use more than 90% of GPU-Util. The resulting pt file after training is 453 MB, significantly larger than our LSTM implementations (5.7 MiB), indicating it has many more parameters than our previous models.

4.5 Training of models

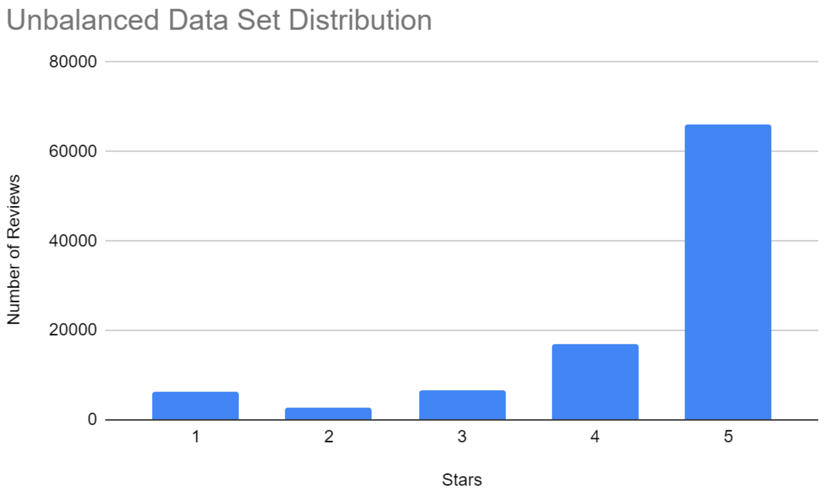
Except for the BERT, we spend time adjusting hyperparameters. It turns out the learning rate is the most important one. For RNN implementations, we manually adjusted the learning rate for every epoch. The analysis below is based on the best test accuracy we are able to achieve.

5 Dataset Details

The dataset used in the seed paper is available online (see references for link). It contains over 5 million product reviews, each of which contains 8 columns; rating, product ID, helpfulness of the review, reviewer ID, reviewer name, review title, date of review, and the review itself. For our project, we kept only the rating and review columns. The number of data entries for each type of rating (i.e. number of stars) was balanced by removing reviews that came from classes that had more reviews. Furthermore, we removed long reviews (i.e. having more than 300 characters). The maximum number of tokens in a review is 226. In order to make use of the review data, we needed to pre-process the text data with tokenization, stop word removal, etc. Finally, the data was split 80%/20% into training and testing data sets.

All of the models were trained on a 500,000 entry subset of the data. Although the data was randomly shuffled before training and testing, the random seed was kept the same between models to ensure consistency.

The BERT model was trained using the balanced data set, but was tested on both a balanced data set and an unbalanced data set consisting of 100,000 entries; 6,350 1-star, 2,883 2-star, 6,695 3-star, 16,891 4-stars, and 66,181 5-star reviews.



6 Evaluation Metrics

To provide a baseline for our model, we implemented several other models in addition to the RNN; a naïve bayes classifier, BERT model, and artificial neural network as discussed in section 4. These models were given the same task of predicting the ratings of a review from its text.

In order to compare the performance of the various models, we looked at the confusion matrix that came from classifying the test data sets. In addition, we computed several statistics such as the accuracy and average absolute error.

$$accuracy = \frac{number\ of\ reviews\ correctly\ classified}{total\ number\ of\ reviews}$$

$$average\ absolute\ error = \frac{\sum_{i} |predicted\ class\ of\ review_i - actual\ class\ of\ review_i|}{total\ number\ of\ reviews}$$

For each model, we also calculated the recall, precision, and F1 scores of each class.

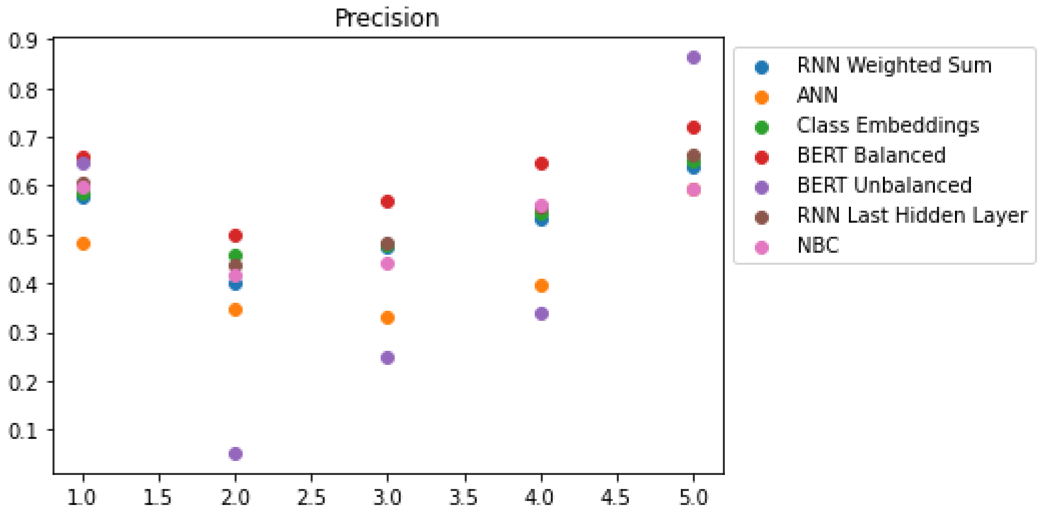
$$precision_i = \frac{number\ of\ class_i\ reviews\ correctly\ classified}{number\ of\ reviews\ classified\ as\ class_i}$$

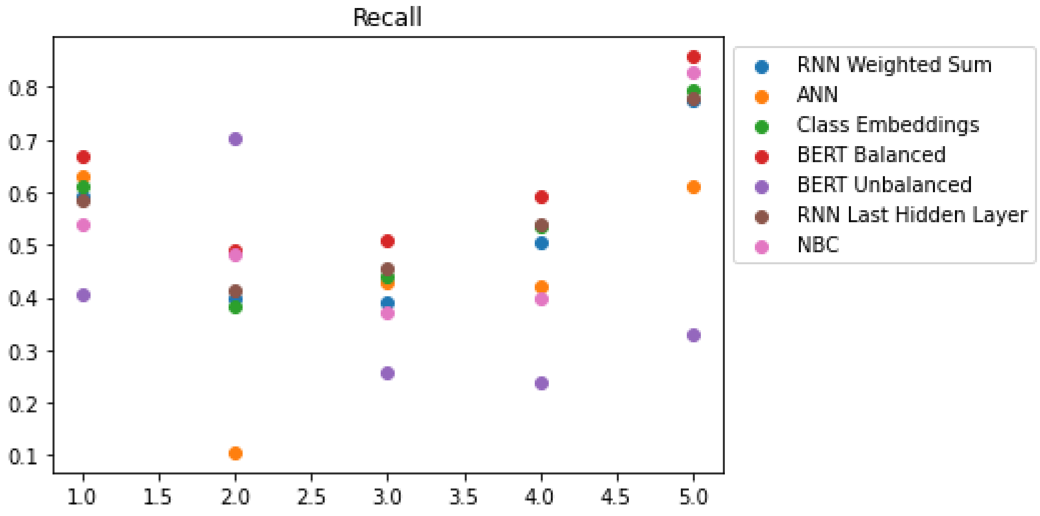
$$recall_i = \frac{number\ of\ class_i\ reviews\ correctly\ classified}{number\ of\ class_i\ reviews}$$

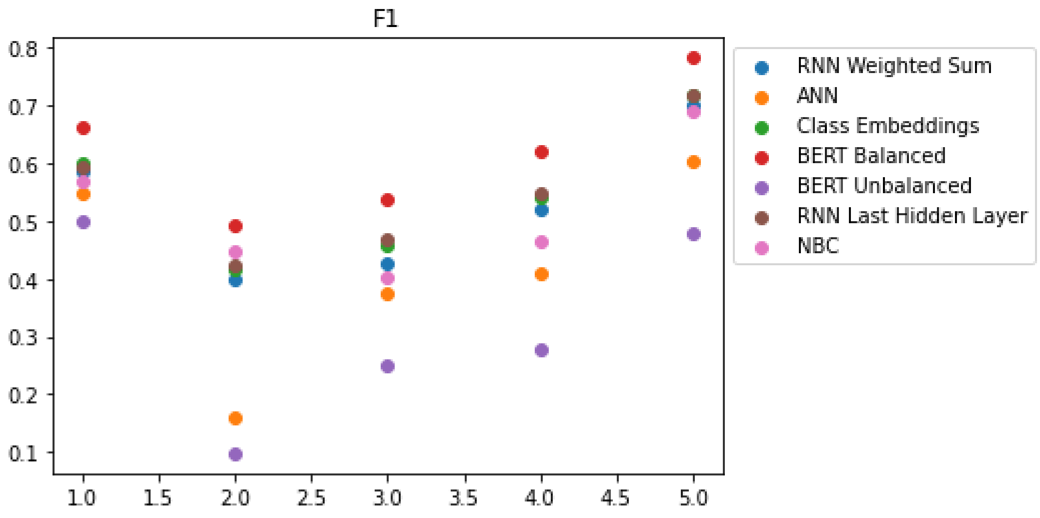
$$F1_i = 2\frac{precision_i \cdot recall_i}{precision_i + recall_i}$$

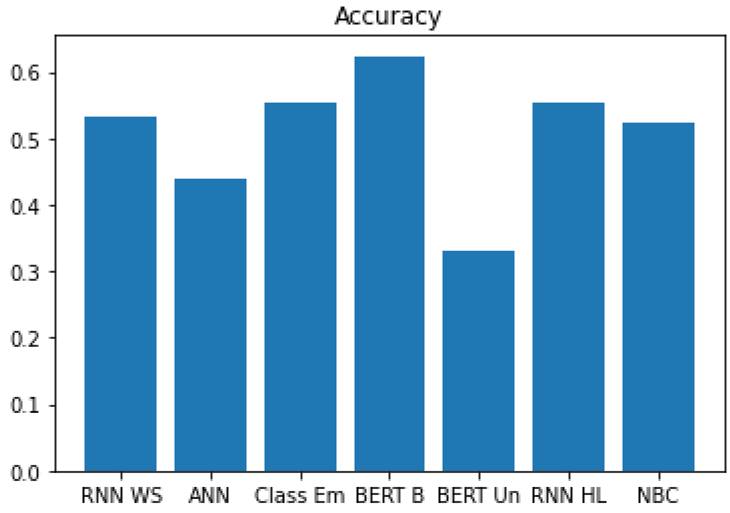
7 Results and Analysis

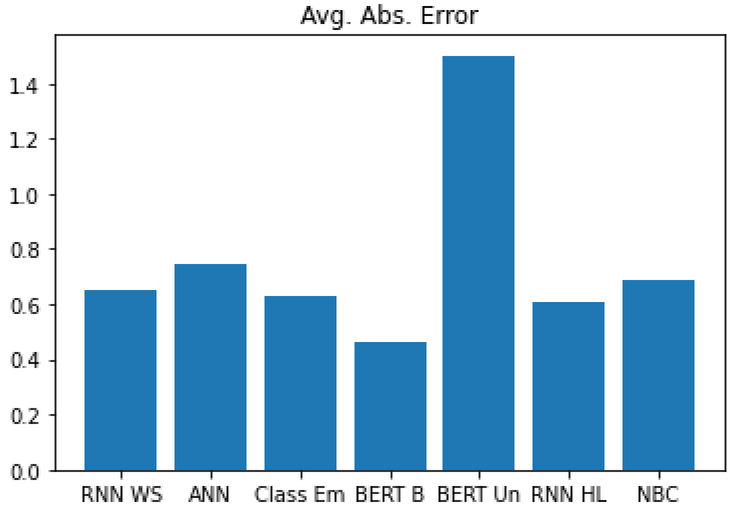
Shown below are graphs that compare the performance of the models used in the study across various metrics. In the precision, recall, and F1 score graphs, each model has five values, one for each class of reviews (i.e. number of stars).






F1
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
● RNN Weighted Sum
● ANN
● Class Embeddings
● BERT Balanced
● BERT Unbalanced
● RNN Last Hidden Layer
● NBC


Accuracy
0.6
0.5
0.4
0.3
0.2
0.1
0.0
RNN WS
ANN
Class Em
BERT B
BERT Un
RNN HL
NBC


Avg. Abs. Error
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
RNN WS
ANN
Class Em
BERT B
BERT Un
RNN HL
NBC

7.1 Analysis of Evaluation Metrics

Overall, BERT performed the worst out of all the models when tested on the unbalanced data set, but performed the best when tested on the balanced data set (all other models tested on balanced data set). We did not spend much time fine tuning the BERT because it takes too much time. However, BERT still achieved the best general performance, indicating the robustness of this state-of-art pre-trained model.

Surprisingly, the naive bayes classifier performed comparably to the deep learning models and even outperformed the ANN.

The poor performance of ANN is expected since it is converting a scalar value to a vector of length five for predictions. The performance of Length of Five Embedding is decent enough given the simplicity of this model. Notice that during training, the overfitting issue does not seem to exist on default learning rate (0.001) for regardless how many epochs.

Both RNN models performed well, though the LSTM with the Last Hidden Layer tended to perform slightly better than the RNN with Weighted Sums. The LSTM weighted sum has similar performance with the Length of Five Embedding baseline. Using the last hidden state of LSTM outperforms that of using the output, indicating that the last hidden state gives better representation on this specific problem.

One trend we noticed with all of the models is that they performed best on 5-star reviews, but worse when predicting 2 and 3-star reviews. The cause of this might lie within the nature of review data. It is easy to state that a highly positive review should be 5 stars, but the distinction between 2, 3, and 4-star reviews is more subjective.

7.2 Weight Analysis

In section 4.1.2, we discussed the weight value for each word which represents its polarity. The top 10 words are 'complements', 'beautiful.', 'cabello', 'clears', 'biting', 'delighted', 'perfect!', 'tweezers.', '"gives', and 'amazing."' The bottom 10 words are 'watered', 'worst', '"worst', 'horrible.', '"broke', '"save', 'hated', 'worst.', '"terrible', and 'awful.' This result indicated this regression task gives reasonable outcomes.

In section 4.2.1, we discussed the weights vector for computing the weighted sum of LSTM outputs. The top 10 words are 'dan', 'printer', 'netflix', 'special.', 'music', 'keyboard.', 'fitbit', 'worse', 'otterbox', 'ruined.' The bottom 10 words are 'it', '', 'of', 'is', 'a', 'i', 'on', 'in', 'N0O0N', 'my.' It makes sense to see a lot of product names with larger weights, because the output layer of LSTM is able to decide the polarity of these products and the larger weight increases the impact of these words to the final estimation of rating. It also makes sense to see all stop words appear with the lowest values. (Note that the “N0O0N” was used to pad sentences which is why it has a low weight.)

7.3 Frequent Words Selection Analysis

For all deep models besides BERT, we found the accuracy of prediction can be increased by ignoring infrequent words. Infrequent words were assigned the index 0, the same value as paddings. We conduct experiments by adjusting the threshold frequency using the LSTM Weighted Sum model. The frequencies we used are all, top 60,000, top 30,000, and top 20,000. The resulting absolute distances are respectively 64152, 63634, 63308, 63611. From here we observed that only keeping part of all tokens generally performs better than when using all of them. Therefore, we end up only using the top 20,000 most frequent words for all RNN models.

7.4 Predictions on Unbalanced Subset

Training on unbalanced sets tended to result in poor performance because 5-star reviews dominated the data. This caused the models to predict that all of the test reviews were 5-stars.

However, we did train a BERT model on a balanced data set and evaluate its performance on an unbalanced data set. (The results can be seen in the graphs above.) In general, the model performed quite poorly when evaluated on the unbalanced test data set.

This may be due to the pattern of our dataset. Our dataset sorts the reviews based on the type of product. The poor result may indicate that the model trained by predicting scores on certain types of product cannot be used to predict other products. For example, cannot train a model using reviews of makeups to predict reviews of foods. This observation has not been verified due to the time limit.

7.5 Error Analysis

Some of the classification errors made by the BERT model were printed out and manually inspected. The largest errors (more than 2 stars), appeared to be caused by mislabelling of the reviews.

For example, here is one such review; "*bought it for my mom who has a lot of pets to help with the hair issues. worked great. she said it actually changed the color of her carpet due to the great suction. she loves it.*" The review was predicted to be five stars, but is labeled as two stars. Here is another example, *"our family loves slim fast, it is a great substitute for breakfast when we are running late. i was so happy to see that i could get this purchased in bulk on amazon. packing was poor and product was askew and dented upon arrival."* This review was predicted to be two stars, but labeled as five stars. Although manually removing mislabeled reviews might improve the model performance, it would be impractical for humans to manually check thousands of reviews.

In this following example, the reviewer used sarcasm in the first sentence which BERT failed to recognize. *"if you have healthy nails this product excellent for you if your nails are not healthily you have problems this products will burn like he!! and leave leave dark marks on your nails. i suggest if you don't have healthy nails try strengthen them frist before using this product."* BERT predicted the review to be 5 stars, but it is actually 2 stars. The model was likely thrown off by the use of the phrase "this product excellent," which was not used sincerely.

8 Conclusions

During this project, we discovered that BERT outperforms the other deep learning models tested on the task of sentiment analysis (when used on a balanced data set). However, it performs significantly worse when used on an unbalanced data set. Given that reviews are generally not balanced in the real world, there is clearly still room for improvements in our implementation.

Also, naive bayes classifiers perform roughly as well as our deep learning models (other than BERT). Given that NBCs are generally less resource intensive than deep learning models, can be trained with less data, and are simpler to implement, they may be a better model choice for some applications.

If we were to continue this project, we would like to try training the deep learning models to take different word embeddings as input (e.g. GloVe and word2vec), instead of the review text data directly, and see how that affects the performance of the models. In addition, we would like to have further explored the performance of the models between balanced and unbalanced data sets.

9 References

Data Set: http://www.itk.ilstu.edu/faculty/xfang13/amazon_data.htm

[1] Seed Paper
https://arxiv.org/ftp/arxiv/papers/1904/1904.04096.pdf

[2] Deep Learning Sentiment Analysis of Amazon.com Reviews and Ratings

https://arxiv.org/pdf/2009.09708.pdf

[3] An Easy Tutorial about Sentiment Analysis with Deep Learning and Keras

https://towardsdatascience.com/an-easy-tutorial-about-sentiment-analysis-with-deep-learning-and-keras-2bf52b9cba91
[4] Sentiment Analysis with Deep Learning
https://towardsdatascience.com/how-to-train-a-deep-learning-sentiment-analysis-model-4716c946c2ea
[5] Deep Learning for Sentiment Analysis: A Survey
https://arxiv.org/ftp/arxiv/papers/1801/1801.07883.pdf

[6] On the Properties of Machines Translation: Encoder-Decoder Approaches

https://arxiv.org/pdf/1409.1259.pdf
[7] Beginner's Guide on Recurrent Neural Networks With PyTorch
https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/
[8] Sentiment Analysis With GloVe
https://www.cs.toronto.edu/~lczhang/aps360_20191/lec/w06/sentiment.html
[9] GloVe + RNN for Sentiment Analysis
https://www.cs.toronto.edu/~lczhang/360/lec/w06/rnn.html