Classifying Amazon Reviews with Star Ratings

Riya Dholakia `driya@seas`

Caroline Liongosari `cliongos@seas`

Sowmya Sridharan `sowmya@seas`

Alicia Yen `aliciayn@seas`

Abstract

Customer product reviews are, and will continue to be, valuable currency for businesses' success and growth. Much research has been done to extract valuable insights from these reviews, such as customer sentiment analysis using multinomial Naive Bayes. This project explores and extends from such models by exploring transformer models such as BERT and its variants, as well as RNN and CNN models, including GRU and LSTM, with the task of using Amazon product reviews to predict 1-5 star ratings. Our results conclude that out of these models, RoBERTa has the highest test performance with a 68.4% accuracy, 0.683 F1 score and a 0.705 RMSE.

1 Introduction

In the new era of online shopping, customer reviews play a crucial role in influencing consumers' buying patterns. Businesses collect customer reviews and extract insights in order to identify product or service shortcomings, assess customer sentiment, improve upon customer service, and more. With the proliferation of social media, product review websites, and online forums, however, it becomes very time-consuming for businesses to manually go through all these reviews to get an understanding of customers' views on their goods and services. Because of this, sentiment analysis is a crucial part of being able to extract insights from hundred thousands to millions of reviews in an efficient manner. It is therefore a fundamental natural language processing task that is continuously being researched in the NLP community.

For our project, we desired to gain a better understanding of this important topic by using Amazon.com customer reviews in order to explore several different classification techniques and text preprocessing methods to predict customers' sentiments. In more specific terms, the goal is that given an Amazon review text for a product from a customer, a model can accurately predict its star rating, which ranges from 1 to 5 stars.

For instance, a classification model would be able to accurately predict a star rating of **5** given this input Amazon product review text: `Great charger. I easily get 3+ charges on a Samsung Galaxy 3. Works perfectly for camping trips or long days on the boat.`

2 Literature Review

For this project, we reviewed several different papers who worked on the same or a very similar classification task and summarize their approaches and results below.

(Taparia and Bagla, 2020) performed the same task that we are exploring: predicting star ratings from Amazon customer review text. They utilized TF-IDF vectorization to transform the review text data into numerical data. They explored 3 classification models: multinomial Naive Bayes, logistic regression, and linear SVC. These models achieved the following accuracy scores respectively: 49.2%, 54.1%, and 51.1%.

(Liu, 2020) did a similar task but used Yelp restaurant reviews rather than Amazon product reviews with the goal of predicting star ratings. They first used different vectorizers to convert the review texts into input for logistic regression and compared the different vectorizers. The different vectorizers had the following settings of setting the minimum document frequency as 5, converting all characters to lowercase, removing all stopwords, and using unigrams and bigrams. The two vectorizers used were sci-kit learn's Countvectorizer and TF-IDF Vectorizer, each using both the binary and integer representations. TF-IDF vectorization with integer representation performed the best with 63.9% accuracy and a 64.3% F1 score, with the worst being count vectorization with binary representation with an accuracy and F1 score of 62.9%.

As such, the TF-IDF vectorization with integer representation was used for the rest of the models except for transformer-based models.

The following non-transformer-based models were used: multinomial Naive Bayes, logistic regression, random forest, and linear SVM. Transformer-based models used (both cased and uncased) were BERT, DistilBERT, RoBERTa, and XLNet. Out of the non-transformer-based models, logistic regression performed the best with an accuracy of 64.1% and 64.5% F1 score. Naive Bayes had the second highest F1 score with 62.2%, then linear SVM with 60.4% and lastly random forest with 59.5% accuracy and 58.7% F1 score. For the transformer-based models, XLNet performed the best with a 70.9% F1 score and 70.4% accuracy, with the worst being uncased BERT with 69.1% accuracy and 69.6% F1 score. All models were better at predicting 1 and 5 star reviews compared to 2, 3, and 4 star reviews. In addition, the paper provided training duration on a GPU. The shortest duration was Naive Bayes at 5 seconds while the longest was XLNet (cased) at 7 hours and 11 minutes.

(Balakrishnan et al., 2021) used deep learning approaches on customer reviews for an e-commerce site selling women's clothing in order to predict star ratings. Unlike the previous two papers, this paper did two different setups for star ratings: 1) using 3 classes ranging from 1 to 3 and 2) using 5 classes ranging 1 to 5. They used word embeddings such as Word2Vec and FastText as features after data preprocessing. They did 3 different experimental scenarios on this classification task: 1) Implementing RNN, CNN, and Bi-LSTM using Word2Vec and FastText 2) BERT variants (BERT, RoBERTa, and ALBERT), and 3) Ensemble Models (CNN-RNN, CNN- Bi-LSTM, RNN-Bi-LSTM, and CNN-RNN-Bi-LSTM). For these models, AdamW was used as the optimizer and CrossEntropyLoss was used as the loss function. 10-fold cross validation, a learning rate of 0.001 and a dropout rate of 0.3 were used.

In the first experiment, Bi-LSTM with Word2Vec performed the best in both the 3-class and 5-class setups with a 63.10 accuracy and 42.59 F-score for the 5-class setup. In the second experiment, RoBERTa performed the best for both the 5-class and 3-class setup with a 69.99 accuracy and 54.79 F1 score for the 5-class setup. In the third experiment, CNN-RNN-Bi-LSTM performed the best with a 96% accuracy and an F-Score of 91.1% on the 3-class setup (no information on the 5-class setup was provided).

3 Experimental Design

3.1 Data

The original dataset for this project was sourced from Kaggle. It contains 10 years worth of Amazon reviews of fine foods up until October 2012 with a total of about 500,000 reviews and is a dataset taken from (McAuley and Leskovec, 2013), who obtained the data by using a webcrawler on Amazon.com directly. However, to make dataset balanced (same number of star ratings for each review) for our use, the dataset was cut down to 140,000 reviews with an 80% training, 10% development, and 10% testing split with each dataset having the same number of 1,2,3,4 and 5 star reviews. The breakdown for each dataset is shown below:

Dataset	**Number of Reviews**
Training Set	112,000
Development Set	14,000
Test Set	14,000

Table 1: Number of Amazon Reviews by Dataset

For each row in the dataset, there are 4 columns: the review id, the star rating of the review (ranging from 1-5), review body text, and the summary (header of the review body). A snippet of one row of data is shown below:

id	**star_rating**	**review_body**	**Summary**
154598	2	bought this recently and will not purchase again. for someone who likes strong coffee this did not do it for me	not strong enough

Table 2: One Row of Data Representing 1 Review

The star_rating is what all classification models will be attempting to predict accurately given the review_body and Summary columns concatenated together.

3.2 Evaluation Metric

For our evaluation script, we provided 3 different metrics for evaluating the performances of our baselines and our future extensions for classifying Amazon product reviews with 1 to 5 star ratings: accuracy score, root mean square error (RMSE), and average F1 score.

Accuracy is defined as: # of predictions exactly matching the true star rating label divided by the total # of predictions. RMSE is defined by the following equation: $\sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2}$ where y_i represents the true/gold standard label while $\hat{y}_i$ is the predicted label. Average F1 is generally defined as $\frac{2*\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$ using the weighted average of the F1 scores for each label.

These are all commonly used metrics for machine learning. All papers listed in the literature review have used the accuracy and F1 score metrics to gauge model performance. We added RMSE as a metric in order to measure how "close" a model gets to the correct answer even if it predicted the wrong star rating. For instance, RMSE gives more credit to a model that predicted a 5 rather than a model that predicted a 1, for a 4-star review. All the other metrics are more simplistic in its scoring system; either the model gets it right or wrong.

3.3 Simple Baseline

Two simple baselines were implemented: average rating and random rating. The average rating baseline calculates the average star rating from the training set and assigns that average star rating to every new review. Since there is a balanced number of classes for the training, development, and testing set, the average rating is 3. As a result, this baseline labels each review in the development and testing set as a 3. The random rating baseline assigns each new rating a random rating from 1 to 5. Below are the results of these baselines on the test set:

	Avg. Rating	**Random Rating**
Accuracy	0.200	0.198
Avg. F1 Set	0.067	0.198
RMSE	1.414	2.006

Table 3: Evaluation Metrics for Simple Baselines

4 Experimental Results

4.1 Published baseline

The published baseline was taken from one of the models implemented in (Taparia and Bagla, 2020). It is the multinomial Naive Bayes classifier using TF-IDF vectorization on the review texts as inputs. TF-IDF provides a better representation of text compared to a CountVectorizer for non-transformer based models since it not only accounts for the number of times a word/token appears in a document, but also considers how often the word/token is across multiple documents, thereby giving lower weights to words/tokens that appear often in general or across documents.

The multinomial Naive Bayes classifier is a well-known classifier for natural language processing. Unlike classifiers such as logistic regression and support-vector machines, the Naive Bayes classifier is a generative model that calculates the joint probabilities of the inputs and the output classes and learns these probabilities using maximum-likelihood estimation on the training data. It also makes the "naive" assumption that all the features are conditionally independent.

This published baseline was implemented using the same methodology as the paper by first using sklearn's CountVectorizer and then using sklearn's TF-IDF transformer to extract the TF-IDF vectors. No data preprocessing such as removing stopwords was done before vectorizing. These TF-IDF vectors were then used as inputs into sklearn's multinomial Naive Bayes classifer. The following results are listed below on the test set:

	Naive Bayes Classifier
Accuracy	0.491
Avg. F1	0.492
RMSE	1.165

Table 4: Evaluation Metrics for Published Baseline

Unsurprisingly, this published baseline performs better than both of the simple baselines in Table 3 with higher accuracy and average F1 scores and a lower RMSE. Despite using a different dataset than (Taparia and Bagla, 2020) since their data set is not publicly accessible, we were able to get a comparable accuracy score of 49.1% vs. their accuracy score of 49.2%.

4.2 Extensions

For our extensions, we extended beyond the published baseline that utilizes the multinomial Naive

Bayes classifier to models that use deep learning. We explored two different areas of deep learning models as our two extensions: BERT and its related models, and RNN and CNN models.

Extension 1: BERT and Related models BERT (Bi-directional Long Short Term Memory) is a transformer model that pre-trains on unlabeled textual data to come up with deep bidirectional representations using both left and right context. It's pre-trained to do masked language modeling and next sentence prediction. The pre-trained model can then be fine-tuned using semi-supervised or supervised training to perform various well-known tasks such as sentiment analysis, disambiguation of words with multiple meanings, and sentence classification. Given that it is currently the start-of-the-art model for many NLP tasks, we wanted to explore how well BERT and its related models can perform on our task. For our implementation of BERT, we used the smaller of the two BERT architectures called base BERT, which has 110 million parameters to perform our task of classifying Amazon reviews to star ratings. For parameter tuning, we experimented with the number of attention heads and the type of position embedding to use.

There are three types of position embeddings: absolute, relative key, and relative key query. Absolute position embedding encodes the absolute position of a token in a sequence (typically ranges from 1 to the maximum sequence length (typically 512). Relative key position embedding encodes the distances every two-token pairing in a sequence. Relative key query extends relative key position embedding by adapting the self-attention mechanism in BERT to incorporate interaction between the relative position embeddings (Huang et al., 2020).

Other than BERT, we also experimented with two other BERT-related models called RoBERTa and DistilBERT. RoBERTa improved on BERT by changing the training methodology such that it retrained BERT using dynamic masking: the masked token changes during the training epochs. It also used significantly more text than BERT for training (16 GB for original BERT vs 160 GB with RoBERTa). DistilBERT focused on inference speed by keeping 97% of the performance of BERT while only using half the number of parameters (66 million parameters vs original 110 million) and reducing the training time to a quarter of the original amount.

Besides experimenting with the various models and their different parameters, we also experimented with different text pre-processing steps including removal of stopwords, numbers, punctuation, HTML tags, contractions, emojis, whitespace, and filtered out words with the part-of-speech tags 'DT', 'PRP', 'CD', 'WDT', 'WP', 'TO', 'IN', 'CC', 'PRP$', and 'WRB' we believed were unimportant for predicting star ratings. We found that using all of these steps led to all models performing the best. The results are on the validation set are shown in Table 5 for some of the models we ran using 2 epochs.

From these results, we find that RoBERTa with absolute position embedding, 12 attention heads, and POS tagging in pre-processing performed the best, having the highest accuracy and F1 score. This made sense since RoBERTa is trained on significantly more data than BERT. We also note that DistilBERT worked as intended by being the fastest model to train at only 2 hours while all other models took 3.5 hours or more. Despite having a lower accuracy, F1, and higher RMSE, the results were not terribly worse compared to the other models, so if speed is more important than getting the absolute highest performing model, DistilBERT is certainly a viable option.

We also found that adding additional attention heads does not necessarily lead to model performance due to redundancy (Michel et al., 2019). In addition, relative key embeddings are typically used for machine translation tasks, not necessarily for classification tasks like we are attempting to do with the Amazon product reviews (Wang et al., 2021).

Extension 2: RNN and CNN models For the second extension, we wanted to explore deep learning models that do not use transformers. As such, we explored two RNN models: LSTM and GRU and their bidirectional counterparts: Bi-LSTM and Bi-GRU. We also explored using a CNN model for this task.

RNN or recurrent-neural network, is a class of deep learning models specifically designed to process sequences of information while retaining the memory of previous inputs. At every stage, the model considers all previous inputs with the current input. It is therefore a model well suited for text classification. There are two main types of RNN models: LSTM (long short-term memory) and GRU (gated recurrent units). These two mod-

Model	# of attention heads	Pos. Embedding Type	Accuracy	Avg. F1	RMSE	Training Time
BERT - cased (no POS tagging)	12	Absolute	0.668	0.663	0.745	3.5 hrs
BERT - uncased (no POS tagging)	12	Absolute	0.665	0.665	0.744	3.5 hrs
BERT - cased	12	Absolute	0.679	0.678	0.723	3.5 hrs
BERT - cased	24	Absolute	0.667	0.792	0.667	4.5 hrs
BERT - cased	12	Relative Key	0.605	0.605	0.924	4 hrs
BERT - cased	12	Relative Key Query	0.610	0.610	0.925	4 hrs
DistilBERT - cased	12	Absolute	0.661	0.661	0.797	2 hrs
RoBERTa - cased	**12**	**Absolute**	**0.680**	**0.678**	**0.739**	**3.5 hrs**

Table 5: Results for BERT and Related Models

els have bidirectional counterparts called Bi-LSTM and Bi-GRU which, unlike their vanilla counterparts, have the ability to read inputs from left to right as well as right to left. These models tend to be more accurate because they process more information in order to make the right prediction. CNN (Convolution Neural Network) is a class of deep learning models typically used for image classification. However, it has capabilities of classifying text as well.

For this extension we first compared different pre-processing methods including removing stopwords, words with certain POS tags, punctuation, etc. Table 6 shows the performance of the Bi-LSTM model using dropout rate of 0.3, learning rate of 0.003, the Adam optimzer, and using Word2Vec as the word embedding method on the product review text. We found that using more text-preprocessing such as removing HTML tags, web addresses, and emojis decreases model performance. This can be explained by the fact that these text attributes actually have relevance in determining the correct star review rating. We also found that using Word2Vec performed the best compared to GloVe and Fasttext, and that the Adam optimizer performed better than SGD with 0.9 momentum and Adagrad for this task.

Using the best pre-preprocessing methods and configurations for word embeddings and the optimizer, we then compared all the different architectures' performances. The results on the development set are shown in Table 7.

Model	Acc.	Avg. F1	RMSE	Training Time
LSTM	0.672	0.671	0.809	12 min
Bi-LSTM	**0.679**	**0.680**	**0.774**	**18 min**
GRU	0.660	0.658	0.787	10 min
Bi-GRU	0.667	0.666	0.783	15 min
CNN	0.627	0.627	0.928	5 min

Table 7: Results for the RNN and CNN architectures

Since CNNs are usually used for image classification, it is not surprising that performed worse in all three evaluation metrics compared to the RNN models. Out of the RNN models, both Bi-LSTM and Bi-GRU outperformed their vanilla counterparts. This is also expected given that the bidirectional models have more information to use to accuracy classify the star rating. Bi-LSTM outperformed Bi-GRU to be the best-performing RNN model. This aligns with the research done in (Yang et al., 2020) which concludes that "GRU performance will surpass LSTM in the scenario of long text and small dataset, and inferior to LSTM in other scenarios." Since our experiments uses short text (less than or equal to 200 characters) and a large dataset of over 2,000 rows, our dataset would not fall into the category that would lead to GRU performing better than LSTM.

Extensions' Results on Test Set Table 8 shows the results the best model from Extension 1, which

Pre-processing Steps	Acc.	Avg. F1	RMSE
lowercase, remove numbers, remove punctuation, expand contractions	**0.6792**	**0.6798**	**0.7744**
lowercase, remove numbers, remove punctuation, expand contractions, remove html tags, web addresses, emojis	0.6786	0.6785	0.7795
lowercase, remove numbers, remove punctuation, expand contractions, lemmatize words	0.6761	0.6766	0.8016
lowercase, remove numbers, remove punctuation, expand contractions, remove html tags, web addresses, emojis, remove stopwords, remove POS tags	0.6536	0.6528	0.9066

Table 6: Results various Pre-processing Methods on Bi-LSTM

is the RoBERTa model, and the best model from Extension 2, which is the Bi-LSTM model. While they both had similar results on the development set, it is clear that RoBERTa outperforms bi-LSTM on the test set by having a higher accuracy and average F1 score and a lower RMSE. Since RoBERTa is considered one of the start-of-the art models that is widely used today, it is not surprising that it will outperform bi-LSTM.

Model	Acc.	Avg. F1	RMSE
Ext. 1: RoBERTa	**0.684**	**0.683**	**0.705**
Ext. 2: Bi-LSTM	0.676	0.677	0.783

Table 8: Results on Best Extensions on Test Set

4.3 Error Analysis

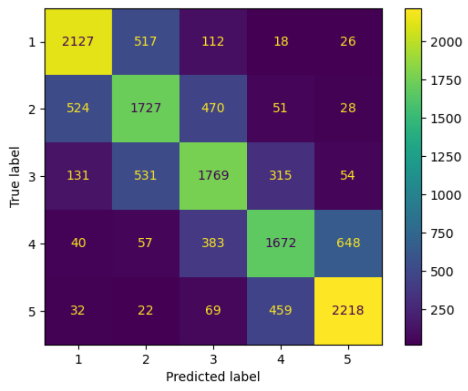

Figure 1: Confusion matrix for Best Model (RoBERTa)

As mentioned in section 4.2, our best model achieved an accuracy score of 68% and an average F1 score of 67.8%. For RoBERTa, we see that most of the misclassified instances occurs for predictions that are within +1/-1 of the true label. For example, most of the errors occur on squares adjacent to the diagonal. The classifier is able to accurately classify 5 star, 1 star, and 3 star reviews best (and in that order). Intuitively, this makes sense that 5 star and 1 star reviews are the two classes that are most accurately classified, as they tend to have stronger language and patterns associated with them due to the polarity of the reviews. Our best model is also able classify neutral (3 star) reviews, though not as well as 1 star and 5 star reviews. This signals to us that RoBERTa is able to classify reviews based on positive, negative, and neutral sentiment, but has problems distinguishing between reviews that are close together. For example, it is a much harder classification problem to distinguish between a 4 star review from a 5 star review, than it is to distinguish between a 1 star review from a 5 star review. The two classes that RoBERTa confused the most were 4 star and 5 star reviews (648 instances were incorrectly predicted as 5 star, when the true label was 4 star).

Reviews that were incorrectly classified by RoBERTa fall into the following broad categories with examples:

1. Mixed messaging between product review and experience review The model makes an error when the review contains mixed messaging between the product review and the experience review (e.g. an item may have arrived damaged or expired). In row 1 below, the actual label of 3 is likely due to the cookies being stale (an issue with storage and QA) and not due to the cookies themselves ("always favorite taste"). However, the language in this review is quite strong ("stale", "disappointed") which led to the model predicting a low negative score.

Review	Actual	Pred.
Stale cookies cookies always favorite taste Famous Amos cookies However ordered amazon bag stale even expiration date taste super dissapointed even prevented ever buying	3	1
Melted could give gift friend loves could give gift melted plastic really stuck carmal bought waybr br dissapointed	3	2
opened package Ordered boxes Jolly Rancher candy box opened sent entired order back never order candy online	2	1

2. Confusion due to possible information loss As mentioned in Section 4.2, text preprocessing can lead to information loss as text attributes are being removed from the review. In row 1 below, the review mentions "amazing" 3 times, along with other positive words like "wow". There is nothing negative indicated in the preprocessed review, which led to the model predicting a high positive rating of 5. However, since the actual label of the review is equal to 3, it is likely that the original review contains information for why the customer gave it a neutral rating.

Review	Actual	Pred.
amazing Have putting beet grill fry amazing Took deer camp last week put fresh deer steaks wow amazing Even better sprinkle little bit salads	3	5
Mocha Capp Breakfast Cookie individually wrapped cookies came good condition Mocha Capp favorite flavor nice change smells great toaster oven	3	4
banana pancake mix love pancakes trying different flavors kept taste buds going	4	5

3. Confusion with classes that share similar sentiment As mentioned at the beginning of the section, the model has the most trouble with predicting classes that are within +1/-1 of the true label (1 star vs. 2 star, 4 star vs. 5 star, etc.). In all examples below, the model was able to accurately capture the sentiment of the review (positive vs. negative) but unfortunately the correct class was not predicted.

Review	Actual	Pred.
husband LOVES coffee Weve tried loved flavored coffees little worried Melitta version would good costs much lessbr br pleasantly surprised coffee smooth rich blend husband loves hazelnut creamer absolutely enamored Creme Brulee Hazelnut flavor Im fan nutty-flavored coffee favorite cant wait try Melitta versions Vanilla Bean	4	5
Really bad coffee bought coffee wouldnt go way go specialized coffee shop Big mistake strong flavorful coffee expect Kona beans taste theyve sitting around months probably reality waste money better Starbucks beans	2	1
Delicious My kids prefer mix regular wheat crust enjoy easy prepare versatile	5	4

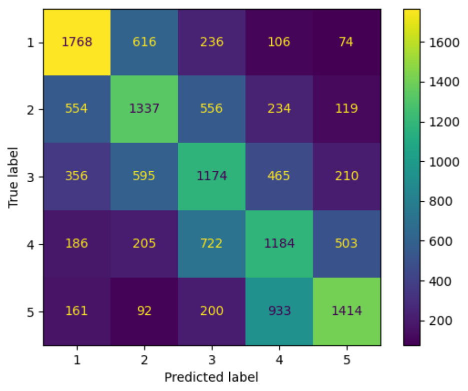

Figure 2: Confusion matrix for published baseline

Our best model performed better than our published baseline, which achieved an accuracy score of 49.1% and an average F1 score of 49.2% (see section 4.1). For Naive Bayes, we see that most of the misclassified instances also occurred for predictions that are +1/-1 of the true label, but we also a significant amount of errors for predictions that are ≥ 2 classes away.

The 1 star review below was misclassified by Naive Bayes as a 4 star review, despite the review containing strong negative language. Our best model was able to correctly classify the review below.

Review	Actual	Pred.
I love Dr Pepper, Mr. Pips and I even like the Wal-Mart brand...diet or not diet. This stuff is terrible, not even close to a generic brand. Made one batch and in the trash it went.	1	4

Considering the entries on the diagonal in Figure 2, Naive Bayes most accurately classifies 1 star reviews, and it less accurately in classifying 5 star reviews. Furthermore, we see that neutral (3 star) reviews has the lowest count among all the entries on the diagonal, which signals that Naive Bayes is not able to accurately classify neutral reviews. The two classes that the published baseline confused the most were 4 star and 5 star reviews (933 instances were incorrectly predicted as 4 star, when the true label was 5 star).

Comparing Naive Bayes to RoBERTa, not only does RoBERTa outperform Naive Bayes on all evaluation metrics, but we see that RoBERTa is able to classify 5 star, 1 star, and 3 star reviews with much higher accuracy. Thus, this signals that RoBERTa has learned to distinguish between positive, negative, and negative sentiment in reviews much better than Naive Bayes is able to.

5 Conclusion

Overall, we were able to improve upon the published baseline of accuracy 49.1% using deep learning methods including RNNs, CNNs, and transformer-based methods like BERT and RoBERTa.

For the RoBERTa model (best model in extension 1), the paper got a 70.0% accuracy while we got a 68.4% accuracy, which provided the best performance out of all models we experimented with. Differences in results can be explained by the different papers we were using and the limited time we had to fine-tune the parameters; it takes about 4 hours for RoBERTa to fully train. For the Bi-LSTM model (best model in extension 2), we were able to reach similar results as the one in (Balakrishnan et al., 2021) where they were able to get a 63.1% accuracy while we got a 67.6% accuracy. Difference in results may have to do with the different datasets we were using.

In conclusion, we can see that with more advanced models, the task of classifying Amazon product reviews with star ratings becomes more and more successful.

Acknowledgements

We would like to thank our TA, Rebecca Iglesias-Flores, for her guidance for throughout the duration of the project as well as Professor Mark Yatskar for introducing us all that the NLP field has to offer.

References

Vimala Balakrishnan, Zhongliang Shi, Chuan Liang Law, Regine Lim, Lee Leng Teh, and Yue Fan. 2021. A deep learning approach in predicting products' sentiment ratings: a comparative analysis. *J. Supercomput.*, pages 1–21.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve transformer models with better relative position embeddings. *CoRR*, abs/2009.13658.

Zefang Liu. 2020. Yelp review rating prediction: Machine learning and deep learning models.

Julian McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. pages 897–908.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *CoRR*, abs/1905.10650.

Ankit Taparia and Tanmay Bagla. 2020. Sentiment analysis: Predicting product reviews' ratings using online customer reviews. *SSRN Electron. J.*

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On position embeddings in BERT. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Shudong Yang, Xueying Yu, and Ying Zhou. 2020. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pages 98–101.