Amazon Review Sentiment Analysis

Tom Donnelly, William Fallon, Dan Gallagher

Abstract

We sought to apply different machine learning models to predict the sentiment of Amazon reviews using only the associated text. We first applied a series of preprocessing methods to clean our data. Next, we implemented a series of simple models, including Naive Bayes, Random Forests, SVM's, and Bigrams, to serve as a baseline for performance. Finally, we implemented and hand-tuned deep learning models, including CNN's, RNN's, and transformers. Once implemented, we varied several model settings, including optimizer choice, product category distribution, and target label scheme, to analyze how our models performed in different scenarios. Our results indicated that deep learning models were significantly better than our baseline at analyzing sentiment and predicting star ratings. Of the deep learning models we trained, BERT_BASE had the best performance.

Introduction

Sentiment Analysis is one of the most obvious examples of the power of modern machine learning. Thanks to modern data storage and the rise of social media, the amount of currently accessible text information in the world is staggering. This presents a plethora of opportunities; one organization may simply wish to perceive its customers views via tweets while another may want to use transcripts of earnings calls used to predict earnings surprises (Gupta & Leung, 2020). Modern natural language processing provides the answer to these problems. Advancements in deep learning provide keys to this massive treasure trove of sentiment information, which was once gated by prohibitive manpower needs.

One of the most obvious utilities of sentiment analysis is the ability to parse product reviews. Understanding what customers have to say about a product is a cornerstone of market research and is essential to both future product development and advertising endeavors. Yet, while a great deal of research has focused on creating new architectures to detect sentiment in product reviews, far less has focused on exploring the data itself and how its various properties might impact different models. We aim to do just that, taking a broader, more exploratory approach to product review sentiment analysis, comparing a variety of models across different subsets of data and target schemes. Furthermore, we take a more hands-on approach to preprocessing-- something a great deal of literature seems to eschew.

Related Prior Work

The Amazon Review Dataset: Researchers have engineered a variety of model architectures and data augmentation procedures to better classify sentiment, often working with the same eight datasets compiled by Zhang, Zhao, & LeCun(2015). The Amazon Product Review dataset, split into both binary and multiclass problems, serves as two of these test sets. It contains millions of Amazon product reviews across 29 categories of products. The existence of its varied product categories and label granularity allows it to be tested as both a binary and non-binary problem, make it appealing for sentiment analysis.

As mentioned, the dataset is typically studied as part of a larger collection. While some have examined general unsupervised data augmentation procedures[11], few thoroughly examined the Amazon Review Dataset as anything beyond a benchmark against which to test. Indeed, if data is cleaned in a supervised manner at all, it is often minor, such as simply lower-casing the text[12][13]. When the goal is to test a specific methodology against benchmarks, this makes sense-- such low-level procedures make it difficult to compare results with other models. We focus on exploration rather than a

single novel methodology, diving deeper into the data itself to see if manual preprocessing based on domain knowledge can help us match or beat potentially more advanced, state-of-the-art models.

Amazon Review Sentiment Methodologies: A variety of neural network architectures have been used to examine Amazon review sentiment. For the most part, recent research has consisted of various flavors of CNNs[10], RNNs (often with LSTM units)[16][17][3], and transformers[15] (often forms of BERT) (see appendix 1 for more detail). While the majority of literature focuses on positive/negative sentiment, Rathor et. al (2018) [2] attempted to detect neutrality as well through weighted unigrams and traditional ml algorithms (the best being an SVM). However, they did not test any neural networks. Shrestha and Nasoz (2019) [3] built an RNN with product category as a feature, inspiring us to examine how different category distributions affect model performance.

Madgrad: In 2021, Facebook released a new optimizer, Madgrad (Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization), which combines several techniques from both Momentum based optimizers and adagrad in an attempt to solve various performance and use case issues of SGD. Particularly, they showed it to perform well in certain NLP cases for both LSTM and BERT models (Defazio & Jelassi, 2021)[14]. However, since it is so new, it has not been explored much in practice, leading us to compare its performance in sentiment analysis across a variety of neural networks.

Formal Problem Setup (T, E, P)

T: For review r, we predict its star rating denoted as $s_i = f(r_i[w, l, c])$, where W is the set of words in the review, c is the context of the review, and l is the length of the review. Of course stars are also determined by a myriad of unknown factors. We propose machine learning models to predict s given r, using some or all of these parameters. We also examine the binary version of the problem, where the star ratings *r* are condensed into binary positive/negative categories and neutral/non-neutral.

E: We use a dataset of Amazon reviews originally curated by McAuley et. al in 2013 and later updated by Ni, Li, and McAuley in 2019. It contains the text of the review, product category, verified information, and the star rating. We will implement several sentiment analysis algorithms and compare performance. We will also explicitly vary several aspects of our models and underlying data, including: number of categories represented, target variable schemes (multiclass vs binary, positive/negative vs neutrality), and optimizers (always at least comparing Adam and Madgrad).

.

P: With TP as true positives, FP as false positives, TN as true negatives, and FN as false negatives, we define the accuracy, precision, recall, and F1 score of our review test set as: **accuracy =** (TP+TN)/(TP+TN+FP+FN); **precision =** TP/(TP + FP); **recall =** TP/(TP+FN); **F1 score** = 2 * recall * precision / (recall + precision). Our main goal will be to maximize accuracy for a holdout dataset of reviews (see appendix 9).

Methods

Preprocessing (for full breakdown see appendices 3-5)

Creating alternate target labels: We took a more granular, exploratory-analysis-based approach to preprocessing than a majority of the literature. Our first goal was to add additional target schemes to the 5-class targets we had. Granular ratings may obfuscate information; different users might have different perceptions of a 4 and 5, for instance, and these perceptions may even change in the same user over time. Thus, we saw merit in converting the problem to a binary positive/negative one to smooth out potential variations in user scale. Our logic was that, while the lines between 1-2 and 4-5 may blur, generally most people could recognize “3” as a relatively neutral point, using that to classify anything

above as positive and anything below as negative. We defined star ratings of 3 as neutral, but excluded them from the positive/negative problem for the purposes of comparing against established benchmarks.

Detecting "bad" reviews: We attempted to identify aspects of our data that might be influencing our star ratings despite not providing information about sentiment. We applied domain knowledge to detect "bad" reviews i.e. bots, trolls, misclicks, etc. This was done by examining the most frequent words relative to various review lengths, removing one-word reviews with only words counterintuitive to their corresponding sentiment (see appendix 2 for word-frequency visualizations).

Concatenating review summaries: Noticing that review summaries often held useful information, we combined them with the text.

Gibberish detection and spellchecking: We also noticed that a great deal of "gibberish" in our reviews, such as the word "aaaaaa." To further weed out bots or otherwise non-informative reviews, we employed a threshold spelling library-based gibberish detector through the enchant library (see appendix 3), removing any records that were over half gibberish. We used the same library to determine when words were likely simply misspelled, and corrected those words with the most likely alternative.

Text cleaning: Next, we cleaned the text to prepare for better compatibility with pretrained embeddings like Glove and Word2Vec. We made all text lowercase, expanded contractions into multiple words, removed links and newline characters, changed "&" to "and" and "_" to whitespace, and removed punctuation we thought might not hold as much semantic value.

Downsampling: Our data was highly imbalanced (favoring 4s and 5s). After splitting the data to have 50,000 test samples, we downsampled the train set so that it was comprised of 50,000 samples from each star rating. We chose downsampling to deal with class imbalance because we already had far more data than we possessed the compute power to train efficiently.

Baseline approaches we compare against: We created four baseline models commonly used in text classification to compare against our more advanced modes: a bag of word Naive Bayes implementation, a random forest with max depth found by grid search, a linear SVM with SGD loss function, and a character bigram naive bayes model. The review text for the models was first transformed into a term-document representation and then TF-IDF weightings were applied.

Implementation Details: We then tested CNNs, RNNs, and Transformers. Common hyperparameters tuned in all three models were dropout proportions, optimizers, and learning rates. The two optimizers we compared were Adam and MADGRAD, a new stochastic optimization method recently developed by Facebook. Hyperparameters were tuned in stages. First, we used nested loops to train the majority of our hyperparameters on a 20,000 sample subset of our data. We then fine-tuned particularly important hyperparameters, such as learning rate and optimizer choice on the full dataset. For all stages, hyperparameters were tested on a single hold out validation set.

CNN's were the first complex models we implemented. We tested a variety of pre-trained word embeddings on our text data, ultimately settling on GloVe.6b.100d as it yielded the best results. We then ran these embedded inputs through a series of convolutional layers. Our final model included 4 convolutional layers with filter sizes of 1, 2, 3, and 4. The output of these convolutional layers was then fed into a final linear layer. We also incorporated dropout at the final step, with the best out of sample performance coming from a dropout of 0.3. The Adam optimizer with a learning rate of .001 was the best performing optimizer/learning rate combination.

We examined a variety of RNN architectures, varying the number and size of recurrent layers, the presence/type of gating unit (GRU vs LSTM), the bidirectionality of said unit, and the presence of a dropout layer before the final linear layer. We trained two of these models on the full data. GRU_RNN had an embedding layer, followed by 2 unidirectional GRU layers with dropout, followed by a fully connected linear layer. BiLSTM replaced the GRU layers with bidirectional LSTM layers and added an additional dropout layer before the fully connected layer. For both RNNs and the CNN, the data was tokenized in a

way that retained punctuation as individual tokens, word vectors in each batch were padded to be the same length, and the embedding layer was pretrained on 6 billion 100-dimensional glove embeddings.

Finally, we used the hugging face transformer framework to finetune BERT-Mini and BERT-Base uncased models for our classification tasks. The model is first loaded with pre-trained embeddings and then trained on our data for the given task. BERT-Mini[8] consists of 4 layers with a hidden layer size of 254 while BERT-Base[9] consists of 12 layers with a hidden layer size of 768. Words were tokenized with the BERT tokenizer to a max length of 128 tokens, then the model was fine-tuned with the optimal hyper-parameter. A batch size of 32 was used for training over 3 epochs on the entire training dataset. Other hyperparameters were kept to the default BERT values.

We used a variety of other techniques to improve our results, such as gradient clipping, which involves capping the size of gradient values, bucketing, dynamically padding our text to reduce the number of pad tokens in a given batch, and saving our final model parameters as their state at the training epoch with the lowest validation loss (in a sense, dynamically tuning the number of epochs).

Experimental Results:

Questions: We aim to answer the following questions: (1) Do our systems classify reviews by sentiment well? How do our different architectures compare with each other? (2) How does the Madgrad optimizer compare to Adam for product review sentiment analysis? (3) Does the number of different product categories represented by the data impact the performance of our models? (4) How do the classifications made for the binary positive/negative problem compare to those for the 5-class problem? (5) Can our models predict neutrality? *Note: Unless specified, all models were trained on data comprising 18 product categories (appendix 4) with a binary positive/negative target label scheme.

	5-Class			Binary Positive/Negative			Binary Neutral/Extreme		
Architecture	**optimizer**	**test_acc**	**test_f1**	**optimizer**	**test_acc**	**test_f1**	**optimizer**	**test_acc**	**test_f1**
Naive Bayes	NA	0.5294	0.4049	NA	0.8655	0.7353	NA	0.6775	0.5523
Random Forest	NA	0.6233	0.4284	NA	0.8794	0.7439	NA	0.7468	0.5967
SVM	SGD	0.6578	0.4332	SGD	0.8872	0.7644	SGD	0.7210	0.5890
Char Bigram	NA	0.5017	0.3753	NA	0.7285	0.6028	NA	0.7305	0.5996
CNN	Adam	0.7572	0.7007	Adam	0.9581	0.9371	Adam	0.8885	0.8283
GRU_RNN	Adam	0.7495	0.6183	Adam	0.9536	0.7947	Adam	0.8699	0.7430
BiLSTM_RNN	Madgrad	0.7656	0.6209	Adam	0.9574	0.8079	Adam	0.8775	0.7520
BERT_MINI	Madgrad	0.7411	0.6058	Madgrad	0.9515	0.8798	Madgrad	0.8874	0.7631
BERT_BASE	Madgrad	0.7689	0.6565	Madgrad	0.9727	0.9269	Madgrad	0.9134	0.8082

General Results: In all settings, our deep learning models significantly outperformed the baseline models. In each experiment, our worst performing deep learning model's out of sample accuracy was at least 10 percentage points higher than our best performing baseline model. Within the deep learning models, BERT_BASE performed the best in each category. BERT_BASE's test set accuracies in the binary positive/negative, binary neutral/extreme, and 5-class classification settings were 97.27%, 91.34%, and 76.89% respectively. All of our neural networks performed comparably to binary state-of-the art benchmarks (https://paperswithcode.com/sota), and actually outperformed 5-class benchmarks. This speaks to the power of domain-knowledge-based preprocessing in sentiment analysis (for caveats and discussion, see appendix 5).

Madgrad vs. Adam: Madgrad and Adam optimizers were tested for each network and each scheme (with the optimal choice shown in Table 1). The two were relatively comparable in most cases (see appendix 6), though Adam seemed to hold the general edge for our CNN and RNNs, while Madgrad performed better for our BERT architectures.

One-Category vs 18-Category Data: We extracted a set of reviews from a single category (electronics), on which we retrained and retested our models. The same number of samples were used for each set. To our surprise, having a broader range of categories to test did not hamper accuracy. The categorical differences may simply not have been significant enough to have much impact, or perhaps the larger variety of data allowed our models to learn a more robust model of general sentiment, even if the underlying distribution to test was commensurately more variable. Either way, the results suggest that our networks have the capacity to handle some amount of variance in the data.

	18 Category		Single Category	
Architecture	**Accuracy**	**F1**	**Accuracy**	**F1**
CNN	0.9581	0.9371	0.9615	0.9388
GRU_RNN	0.9536	0.7947	0.9535	0.9404
BiLSTM_RNN	0.9574	0.8079	0.9500	0.9367
BERT_MINI	0.9511	0.8765	0.9499	0.9474
BERT_BASE	0.9727	0.9269	0.9645	0.9626

Target scheme differences: As expected, our models performed best on the simple positive vs. negative classification problem and performed worse as we added additional classes. Aside from the fact that more labels make classification more difficult, our context makes this even more of a challenge due to the varying internal rating scales both among users. In practice, it may be best to simplify and combine granular ratings as much as is reasonable. Neutrality was more difficult to classify than positive/negative sentiment. Non-neutral cases will likely have more variability among reviews than positive or negative cases. The neutral case is likely not always truly neutral, or may even be a mix of extreme positive and negative views that "balance out." Nevertheless, our models, particularly BERT_BASE, performed admirably, demonstrating the ability of language embeddings and deep learning to learn a variety of contexts from the same data, especially when compared to "traditional" learning methods which performed fairly well on the positive/negative problem but relatively poor on the neutrality problem.

Conclusions and Future Work

Our complex models were successful in classifying the sentiment of Amazon reviews, significantly outperforming our baseline models and meeting or in some cases exceeding state of the art performance. We attribute some of this success to the extensive domain-specific preprocessing methods we incorporated, something we found uncommon in the literature, which generally focuses more on generalizing a methodology to multiple datasets. In practice, domain-based "manual" preprocessing may still have an important place in the sentiment analysis pipeline. We also demonstrated the difficulty of classifying sentiment analysis from granular data as well as the relative adaptability of embeddings and neural networks to fit a variety of contexts present within the same text. Finally, we found that Facebook's Madgrad optimizer may be a useful alternative to Adam for training BERT models.

Time and resource constraints limited the depth of our experiments; we shallowly examined multiple settings rather than go in depth on any. Future work should compare models across multiple subsets of categories for a more robust examination. Additionally, a 3-class sentiment scheme should be explored (including "3" as neutral) to further tease out the interplay between granularity and predictability.

Ethical Considerations and Broader Impacts:

With any NLP task, implicit biases and generalizations in the dataset could be further exaggerated by our models- for example predicting a more negative sentiment when words related to specific genders or ethnicities that are present in the review. It should also be considered whether these models could be used to imply a wrong sentiment to text, and what action was taken because of it. While we did not examine these issues given the exploratory nature of our project, it is important to ensure that these biases are not present in models that would be deployed and affect real people.

Prior Work / References:

1. McAuley, J., Targett, C., Shi, Q., & Van Den Hengel, A. (2015, August). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 43-52).
2. Jianmo Ni, Jiacheng Li, Julian McAuley. Empirical Methods in Natural Language Processing (EMNLP), 2019
3. Abhilasha Singh Rathor, Amit Agarwal, Preeti Dimri, Comparative Study of Machine Learning Approaches for Amazon Reviews, Procedia Computer Science, Volume 132, 2018, Pages 1552-1561, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.119. (https://www.sciencedirect.com/science/article/pii/S1877050918308512)
4. Shrestha, N. and Nasoz, F., "Deep Learning Sentiment Analysis of Amazon.com Reviews and Ratings", 2019. https://arxiv.org/abs/1904.04096
5. Wang, X., Jiang, W., & Luo, Z. (2016, December). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 2428-2437). https://www.aclweb.org/anthology/C16-1229.pdf
6. Gupta, T. and Leung, E., "Invesco's Guide To Alternative Data", 2020. https://www.invesco.com/content/dam/invesco/apac-master/en/pdf/apac/2020/solutions/Q4-Risk-and-Reward.pdf
7. Timmaraju, A., & Khanna, V. (2015). Sentiment analysis on movie reviews using recursive and recurrent neural network architectures. *Semantic Scholar*, 1-5. https://cs224d.stanford.edu/reports/TimmarajuAditya.pdf
8. Turc, I., Chang, M.W., Lee, K., & Toutanova, K. (2019). Well-Read Students Learn Better: On the Importance of Pre-training Compact Models*. arXiv preprint arXiv:1908.08962v2* .
9. Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. CoRR, abs/1810.04805.*
10. Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*.
11. Xie, Q., Dai, Z., Hovy, E., Luong, M. T., & Le, Q. V. (2019). Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.
12. Johnson, R., & Zhang, T. (2016, June). Supervised and semi-supervised text categorization using LSTM for region embeddings. In *International Conference on Machine Learning* (pp. 526-534). PMLR.
13. Johnson, R., & Zhang, T. (2017, July). Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 562-570).
14. Defazio, A., & Jelassi, S. (2021). Adaptivity without Compromise: A Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization. *arXiv preprint arXiv:2101.11075*.
15. B. Myagmar, J. Li and S. Kimura, "Cross-Domain Sentiment Classification With Bidirectional Contextualized Transformer Language Models," in *IEEE Access*, vol. 7, pp. 163219-163230, 2019, doi: 10.1109/ACCESS.2019.2952360.
16. Yu, Z., & Liu, G. (2018). Sliced recurrent neural networks. *arXiv preprint arXiv:1807.02291*.
17. Minaee, S., Azimi, E., & Abdolrashidi, A. (2019). Deep-sentiment: Sentiment analysis using ensemble of cnn and bi-lstm models. *arXiv preprint arXiv:1904.04206*.

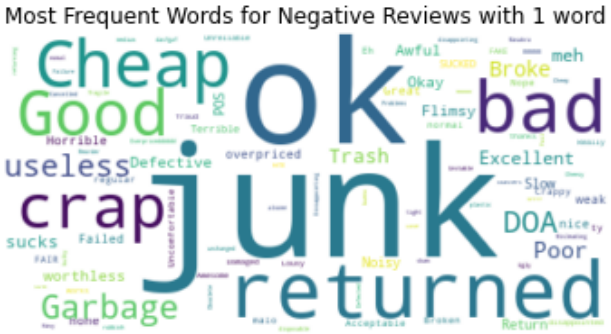
Supplementary Material:

1. More information on deep learning for sentiment analysis:

The sentiment analysis literature becomes more diverse when we expand beyond Amazon reviews. Wang, Jang, and Luo (2016) [4] combined a CNN and RNN together in an attempt to classify sentiment of multiple datasets, using the theory that the CNN would best capture the local features and the RNN (which used either an LSTM or GRU) would better capture longer term sequential information. They found that the combined model performed better than either individually across all datasets. Timmaraju and Khanna (2015) [6] also used a combination of architectures, this time combining Recursive and Recurrent Neural Networks to classify sentiment for movie reviews. While the combination outperformed each individual network, they were interestingly all outperformed by a support vector machine with hand-defined features, suggesting that not every situation necessarily calls for deep learning.

Minaee et.al [17] discuss the applications of CNNs and LTSM models to sentiment analysis. They note similar accuracies for CNN's and LTSM's and also observed a slight increase in accuracy with an ensemble of the two. Myagmar et. al [15] outline the current standard for the use of Transformers in sentiment classification, focusing their analysis on the BERT and XLnet models.

2. Top 100 word frequencies of 1-word reviews:



3. Gibberish detection and spellchecking- the Enchant library:

Enchant combines multiple pre-loaded dictionaries with a variety of third-party spell-checking libraries. These libraries allow for a variety of spell-check methods/resources, such as n-gram similarity, rule and dictionary based pronunciation data, morphological analysis, stemming and generation. Gibberish is defined as words that do not have a probable alternative.

4. Choosing product categories:

The original dataset had a vast range of review counts per category. To prevent questions of dominant categories from muddling comparisons of multi-category and single-category data subsets, we randomly sampled from each product category such that each had the same number of samples before joining. Categories were excluded if they possessed too few reviews to be equally represented with the others. The use of electronics as the single-category subset was determined randomly. Reviews on electronics were also included in the 18-category set.

5. State of the Art Caveats:

While our neural networks performed comparably to binary state-of-the art benchmarks (https://paperswithcode.com/sota), and actually outperformed 5-class benchmarks, there are a number of caveats associated with this. The first, which may not even be a caveat so much as a central driver, is that the SOTA models are often meant for generalized sentiment analysis not just restricted to Amazon reviews, leading our models to be far more specialized for this specific task due to domain specific pre-processing.

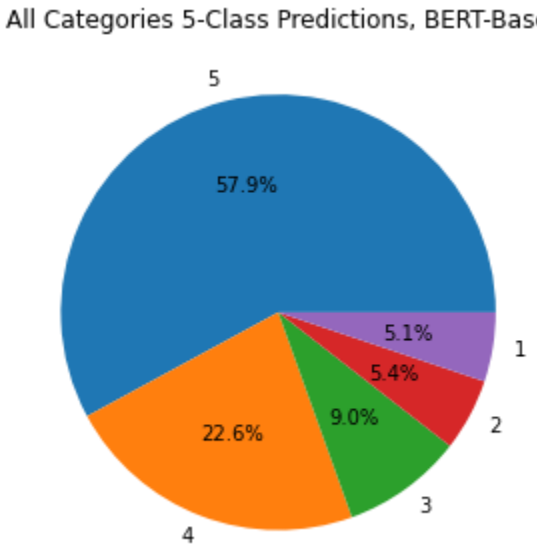
Secondly, our models are trained and tested on a small subset of the 20 million+ reviews in the full corpus while the SOTA models are trained and tested are the entire dataset, thus we cannot say for certain that our models outperform until tested on a larger sample. Additionally, many of the SOTA benchmarks use the Amazon Review Dataset from 2013, before it was updated with more recent reviews in 2018. Thus, our underlying data, while contextually very similar, comes from a different distribution than some of the papers against which we are comparing.

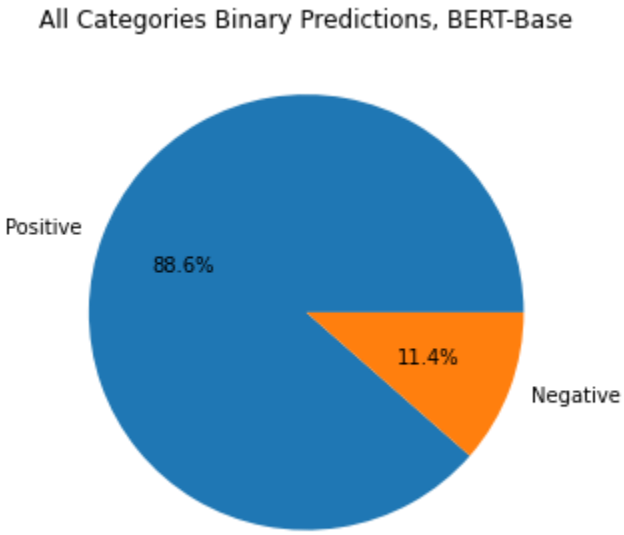
Finally, our preprocessing involves pruning reviews we found to be obviously erroneous from both the train and test sets. This tactic's usefulness (and veracity) depends on the use case. In situations where we incur great cost from saying "this item will not be predicted as anything because it failed x filter," we would not want to test on a set with any records removed, and might not want to use filter-based record removal at all. However, sentiment analysis typically uses static (non-streamed) and abundant data that are analyzed in bulk to gain a greater understanding of feelings. In many cases, little is lost by flagging certain reviews as "non-predictable" by our model, and often more is gained by purging these "bad" reviews before aggregating and analyzing. Nevertheless, no SOTA model we saw removed this type of data (for reasons now touched on many times in this paper), and so their test distribution is not only different, but likely demonstrably harder to predict. Thus, while in practice, we believe removing certain records was the right call, it still served to inflate our accuracy relative to SOTA benchmarks.

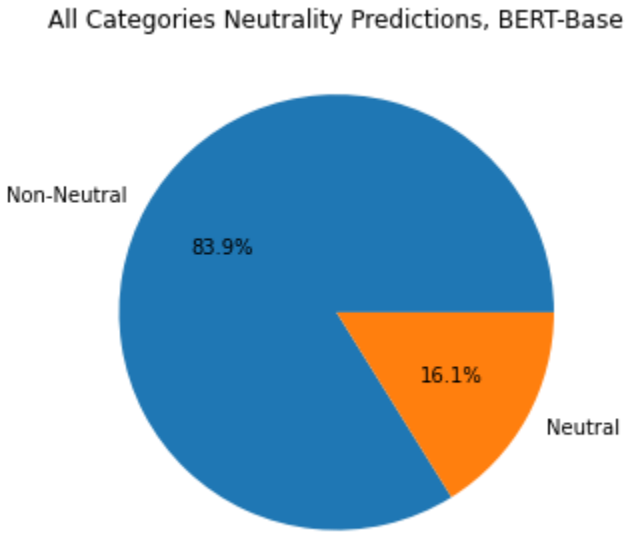
6. Madgrad vs. Adam, all differences (binary positive/negative and 5 class):

Madgrad vs Adam						
Target scheme	**Architecture**	**optimizer**	**test_acc**	**test_f1**	**Recall**	**Precision**
binary	CNN	Adam	0.9581	0.9371	0.9477	0.9318
binary	CNN	Madgrad	0.8820	0.8304	0.8677	0.8212
five_class	CNN	Adam	0.7572	0.7007	0.7208	0.7147
five_class	CNN	Madgrad	0.6451	0.5741	0.5890	0.5893
binary	GRU_RNN	Adam	0.9536	0.7947	0.9570	0.6795
binary	GRU_RNN	Madgrad	0.9410	0.7529	0.9581	0.6201
five_class	GRU_RNN	Adam	0.7495	0.6183	0.6647	0.5857
five_class	GRU_RNN	Madgrad	0.7429	0.6097	0.6581	0.5810
binary	BiLSTM_RNN	Adam	0.9574	0.8079	0.9547	0.7002
binary	BiLSTM_RNN	Madgrad	0.9359	0.7376	0.9593	0.5991
five_class	BiLSTM_RNN	Adam	0.7159	0.6033	0.6506	0.5789
five_class	BiLSTM_RNN	Madgrad	0.7656	0.6209	0.6534	0.5961
binary	BERT_MINI	Adam	0.9460	0.8667	0.9391	0.8201
binary	BERT_MINI	Madgrad	0.9515	0.8798	0.9525	0.8324
five_class	BERT_MINI	Adam	0.7292	0.5895	0.6477	0.5548
five_class	BERT_MINI	Madgrad	0.7411	0.6058	0.6619	0.5704
binary	BERT_BASE	Adam	0.9061	0.4754	0.5000	0.4531
binary	BERT_BASE	Madgrad	0.9727	0.9269	0.9690	0.8935
five_class	BERT_BASE	Adam	0.6368	0.5159	0.5771	0.4967
five_class	BERT_BASE	Madgrad	0.7756	0.6436	0.6796	0.6170

7. Bert-Base Category Distribution







Our model proportions match generally well, there are 11.4% proportional negative reviews to positive review without 3 stars, and this is what our binary model predicts. Additionally there are 87.1% Non-Neutral reviews to 12.9% neutral reviews in our 5-class predictions(we don't count 4 and 2 as non-neutral). Our model predicts 83.9% non-negative to 16.1% neutral reviews, a discrepancy of 3.2%. The fact that our class-prediction proportions generally match well lends credence to the idea that a great deal of the difficulty in going from 2 to 5 classes lies in the potentially blurry lines between the sentiment meant behind adjacent star ratings.

8. Hyper Parameter Table:

Architecture	dataset	prediction_type	optimizer	test_acc	test_f1	test_recall	test_precision	learning rate	dropout	hidden_size
BERT_BASE	Allcats	Five_Class	ADAM	0.6368	0.5159	0.5771	0.4967	5.00E-05	0.1	NA
BERT_BASE	Allcats	Five_Class	MADGRAD	0.7689	0.6565	0.6993	0.6272	5.00E-05	0.1	NA
BERT_BASE	Allcats	Binary	ADAM	0.9061	0.4754	0.5000	0.4531	0.0001	0.5	NA
BERT_BASE	Allcats	Binary	MADGRAD	0.9727	0.9269	0.9690	0.8935	0.0001	0.1	NA
BERT_BASE	Allcats	Neutrality	ADAM	0.9112	0.8051	0.8989	0.7574	1.00E-05	0.5	NA
BERT_BASE	Allcats	Neutrality	MADGRAD	0.9134	0.8082	0.8988	0.7609	0.0001	0.1	NA
BERT_BASE	Electronics	Five_Class	MADGRAD	0.7111	0.6971	0.7006	0.6969	5.00E-05	0.1	NA
BERT_BASE	Electronics	Binary	ADAM	0.3853	0.2781	0.5000	0.1926	5.00E-05	0.1	NA
BERT_BASE	Electronics	Binary	MADGRAD	0.9645	0.9626	0.9644	0.9610	0.0001	0.1	NA
BERT_MINI	Allcats	Five_Class	ADAM	0.7292	0.5895	0.6477	0.5548	5.00E-05	0.1	NA
BERT_MINI	Allcats	Five_Class	MADGRAD	0.7411	0.6058	0.6619	0.5704	5.00E-05	0.1	NA
BERT_MINI	Allcats	Binary	ADAM	0.9460	0.8667	0.9391	0.8201	0.0001	0.5	NA
BERT_MINI	Allcats	Binary	MADGRAD	0.9515	0.8798	0.9525	0.8324	0.0001	0.1	NA
BERT_MINI	Allcats	Neutrality	ADAM	0.8686	0.7365	0.8462	0.6960	0.0001	0.5	NA
BERT_MINI	Allcats	Neutrality	MADGRAD	0.8874	0.7631	0.8621	0.7195	0.0001	0.1	NA
BERT_MINI	Electronics	Five_Class	ADAM	0.6615	0.6469	0.6526	0.6488	5.00E-05	0.1	NA
BERT_MINI	Electronics	Five_Class	MADGRAD	0.6729	0.6572	0.6618	0.6565	5.00E-05	0.1	NA
BERT_MINI	Electronics	Binary	ADAM	0.9398	0.9371	0.9422	0.9332	5.00E-05	0.1	NA
BERT_MINI	Electronics	Binary	MADGRAD	0.9499	0.9474	0.9503	0.9449	0.0001	0.1	NA
BiLSTM_RNN	Allcats	Binary	ADAM	0.9574	0.8079	0.9547	0.7002	0.0005	0	300
BiLSTM_RNN	Allcats	Binary	MADGRAD	0.9359	0.7376	0.9593	0.5991	0.0005	0	300
BiLSTM_RNN	Allcats	Five_Class	ADAM	0.7159	0.6033	0.6506	0.5789	0.0002	0	300
BiLSTM_RNN	Allcats	Five_Class	MADGRAD	0.7656	0.6209	0.6534	0.5961	0.0002	0	300
BiLSTM_RNN	Allcats	Neutrality	ADAM	0.8775	0.7520	0.8640	0.7087	0.0002	0	300
BiLSTM_RNN	Allcats	Neutrality	MADGRAD	0.8441	0.7135	0.8515	0.6774	0.0002	0.5	300
BiLSTM_RNN	Electronics	Binary	ADAM	0.9500	0.9367	0.9596	0.9148	0.001	0.3	200
GRU_RNN	Allcats	Binary	ADAM	0.9536	0.7947	0.9570	0.6795	0.001	0.3	300
GRU_RNN	Allcats	Binary	MADGRAD	0.9410	0.7529	0.9581	0.6201	0.0005	0	300
GRU_RNN	Allcats	Five_Class	ADAM	0.7495	0.6183	0.6647	0.5857	0.0005	0.5	300
GRU_RNN	Allcats	Five_Class	MADGRAD	0.7429	0.6097	0.6581	0.5810	0.0002	0.5	300
GRU_RNN	Allcats	Neutrality	ADAM	0.8699	0.7430	0.8619	0.7008	0.0002	0.5	300
GRU_RNN	Allcats	Neutrality	MADGRAD	0.8559	0.7258	0.8540	0.6868	0.001	0.5	300
GRU_RNN	Electronics	Binary	ADAM	0.9535	0.9404	0.9518	0.9293	0.001	0.3	300
CNN	Allcats	Binary	ADAM	0.9581	0.9371	0.9477	0.9318	0.001	0.3	NA
CNN	Allcats	Binary	MADGRAD	0.8820	0.8304	0.8677	0.8212	0.001	0.3	NA
CNN	Allcats	Five_Class	ADAM	0.7572	0.7007	0.7208	0.7147	0.001	0.3	NA
CNN	Allcats	Five_Class	MADGRAD	0.6451	0.5741	0.5890	0.5893	0.001	0.3	NA
CNN	Allcats	Neutrality	ADAM	0.8885	0.8283	0.8537	0.8236	0.001	0.3	NA
CNN	Allcats	Neutrality	MADGRAD	0.8052	0.6935	0.7348	0.7002	0.001	0.3	NA
CNN	Electronics	Binary	ADAM	0.9615	0.9387	0.9458	0.9358	0.001	0.3	NA
CNN	Electronics	Binary	MADGRAD	0.9099	0.8665	0.8722	0.8687	0.001	0.3	NA

9. A note on Accuracy vs F1:

We used accuracy as a measure of how well our models are doing generally on predicting the correct results. F1 is used in our case to also determine that our models are correctly predicting values across classes, that is that there is not one or multiple classes that are constantly being mislabeled. This is useful over accuracy when the test dataset has an imbalance of labels, as is the case with our Amazon reviews (which are skewed left). As our accuracy and F1 score generally correlate we can have higher confidence in our models. In general, we discussed accuracy over F1 in our paper so that we could better compare to SOTA benchmarks, which also use accuracy.