Predicting Helpfulness of Amazon Reviews

Aditya Mishra

Abstract

This project explores a subset of the Amazon review dataset and makes sure that we have had. Using some manual features and some word embedding techniques, we trained a logistic regression model and a simple fully-connected neural network to help determine whether we can effectively predict how helpful a rating will be. We found out that although the neural network and logistic regression model perform similarly on the test data, the neural network is much more prone to overfitting.

1 Introduction

1.1 Background and Research Question

Many large consumer tech companies such as Amazon and Netflix rely on customer ratings for their recommendation systems, so that they can help other potential customers buy products that they might enjoy. Amazon in particular allows its buyers to leave detailed reviews of various products. Promoting helpful reviews is beneficial to Amazon, as it helps potential buyers not be scammed and have an overall positive experience shopping at Amazon. This begs the question: Can we build effective models that reliably predict how helpful a review is going to be?

2 Dataset

There were multiple versions of the Amazon Dataset. We initially used the newest version released in 2018 (Ni et al., 2019), but subsequently decided to work with the dataset that was released in 2014 (McAuley et al., 2015). Please see Section 2.1 for more details. Due to computational constraints, we also decided to mainly focus on a subset of the full dataset, namely the reviews from the “Digital Music", “Toys and Games" and “Musical Instruments" categories.

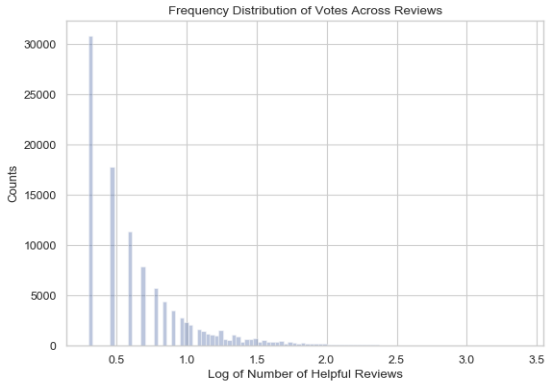

Figure 1: Frequency distribution of the number of helpful votes a review recieved on a logarithm scale.

2.1 Exploratory Data Analysis

From the 2018 Amazon Dataset We explored the distribution of the number of helpful votes. From Figure 1 we see that most of the reviews recieved less than 10 votes, but there are a good amount of outliers recieving well over 50 votes. Formulating a regression problem seems rather difficult, given the vast discrepancy between the number of helpful votes that a dataset got. This discrepancy can potentially be explained by the fact that products differ in popularity. We realized that there was an older version of the dataset that allowed us to normalize these scores, since it provided us the the number of positive *and* negative votes a reviewer has received. As such, we decided to use this dataset.

2.2 Problem Statement

Given a review, we have access to compute a normalized helpfulness score:

$$s = \frac{f_p}{f_{\text{tot}}}$$

Note that $0 \leq s \leq 1$, which seems a lot more friendlier. Now, according to figure ..... the distribution of scores is basically skewed towards the right. We define our notion of a good review as one that falls in the top 50% of our normalized helpfulness score. For the reviews in the "Digital Music" category, we determined hat cutoff to be at 0.8. Our motivation for classifying helpfulness scores as good or bad is to reduce the effect of noise in our training data. It is unlikeley that there is a meaningful difference in a review that has a helpfulness score of 0.66 versus a review that has a helpfulness score of 0.61. Forming general categories will make our classification task easier and focus on general trends in the review data.

2.3 Initial Preprocessing

From the 2014 version of the dataset, we immediately decided to drop the following columns from consideration of our feature space:

- `reviewerID, reviewerName`: This is because we want to make sure that our models only take into account the content of the review instead of making sre. Also with people being more conscious of their digital footprint, we want our models to make sure that we don't rely on user-specific data.
- `asin`: We feel that a useful review is independent of the specific product that it is actually reviewing. We don't want to introduce bias into our model from other factors about our product (i.e. it's popularity).
- `reviewTime, unixReviewTime`: The time feature seems irrelevant to the content of our reviews. Also, if our model is being used in downstream tasks (i.e. making future predictions), we don't want our model to be biased towards a certain time.

The features that were left over were the review summary, the actual review itself, and the star rating.

3 Feature Engineering

We broadly categorize the features we used into three groups: punctuation features that deal with how punctuation is used in the summary and the actual review, word embedding features, and readability metrics.

3.1 Punctuation Features

This group of features is the most simple. We simply counted the number of occurrences of `@, .,  , #` in the review body and text.

3.2 Word Features

We came up with three approaches for our word embeddings.

3.2.1 Count Vectors

We form count vectors of the 300 most frequently used words in the training set for each set of reviews.

3.2.2 TFIDF embedding

We used term frequency–inverse document frequency values to featurize the review text of each entry in our dataset (Leskovec et al., 2014). Given a corpus $C = \{d_1, \ldots d_{n_C}\}$ consisting of n_C documents and words $\{w_1, \ldots, w_V\}$, we define the tfidf score for a particular word i in document j as:

$$\text{tfidf}(w_i, d_j) = \text{tf}(w_i, d_j) \cdot \text{idf}(w_i, d_j)$$

The term-frequency is defined as:

$$\text{tf}(w_i, d_j) = \frac{f_{ij}}{\max_k f_{kj}}$$

where f_{ij} is the number of times a word appears in a document.

The inverse-document frequency is defined as

$$\text{idf}(w_i, d_j) = \log\left(\frac{n_C}{g_{ij}}\right)$$

where g_{ij} is the number of documents that w_i appears in. We only included the 300 words with the highest tfidf scores in our dataset.

3.2.3 word2vec based Document Embeddings

A common method to learn word embeddings is the word2vec algorithm. The word2vec is a 2 layer neural network that is trained on a corpus of text to learn effective word embeddings. It usually comes in two flavors: the continuous bag-of-words model and the skip-gram model. As we used the skip-gram model (Mikolov et al., 2013a), we will describe the latter in this article. At a high level, a given a word sequence $w_1, \ldots, w_T$, the skip-gram tries to maximize the log-probability:

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log \Pr(w_{t+j} | w_t)$$

Here c is defined as the context size, which isn't necessarily constant. We used a pretrained word2vec model trained on a subset of Google News Data (Mikolov et al., 2013b). The pretrained embeddings that we used can be dowloaded here. These pretrained embeddings generate a 300 dimensional embedding for a word.

However, this model can only embed a single word at a time. There are many possible mathematical operations that could be used to combine multiple word-embeddings from a document into a single embedding vector (Conneau et al., 2017). In our work, given word embeddings $e_1, \ldots, e_D$ for a single document, we computed the mean embedding as our final document embedding, $\frac{1}{D}\sum_{i=1}^{D} e_i$.

4 Models

We used two models to classify our reviews as "helpful" and "not helpful". We describe how we trained them in this section.

4.1 Logistic Regression Model

The first model that we used is a logistic regression model. This model is well suited for binary classification tasks and is desirable as it outputs a probability that a document is going to be helpful or not.

4.1.1 Training Details

We use cross-validation to tune the ℓ-2 regularization parameter over the set of values $C \in [10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}]$. The number of folds we use is 5, and we make sure that we preserve the percentage distribution of each class when generating the train and test folds.

4.2 Neural Network Model

We use a shallow neural network . In particualar we use a 2-layer fully-connected neural network with a sigmoid activation function after the last layer to output a probability that a review is helpful. The sizes of our hidden dimensions are 1024 and 256 respectively.

4.2.1 Training Details

We use an Adam optimizer with learning rate 0.01 and used the default parameters provided in the Pytorch implemenation.

5 Results and Discussion

To this end, we take two different approaches to developing models for these classification tasks and evaluate them by their classification accuracies and areas under the curve (AUC) on a test set. From Figure 2, we see that across both models, using simple word counts proved to be worse models than both the tfidf and the pretrained embeddings.

One interesting thing that we found is that the neural network doesn't seem to be outperforming the logisitc regression model. We wondered if this was a potentially due to overfitting. We see that from Figure 3, that the neural network model significantly outperforms on the training set versus the test set. We can therefore conclude that the neural net has trouble generalizing well.

6 Future Work

There are plenty of interesting directions that we can take. We broadly outline these directions in this section.

6.1 Generalizing our Model to Multiple Categories

Large transformer based models with billions of parameters have been trained and can be used to achieve state-of-the-art performance on a multitude of NLP tasks, including text classification (Yang et al., 2019). In our work, we trained data on a small set of categories. One moreaven

6.2 Performing a more aggressive feature selection

While we proposed various features on the composition of our models, we did not conduct any experiments to decide what features our models think are imporant. Unfortunately, the models we chose (logistic regression and a neural network) don't lend themselves to easy interpretation.

7 Conclusion

In this analysis, we present a classical and deep learning approach for classifying the helpfulness of Amazon product reviews. We show that the deep learning model is prone to overfitting compared to the logistic regression model. We explain the rationale for the different parts of our design, discuss trade-offs we encountered, and leave several avenues open for future work.

References

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised

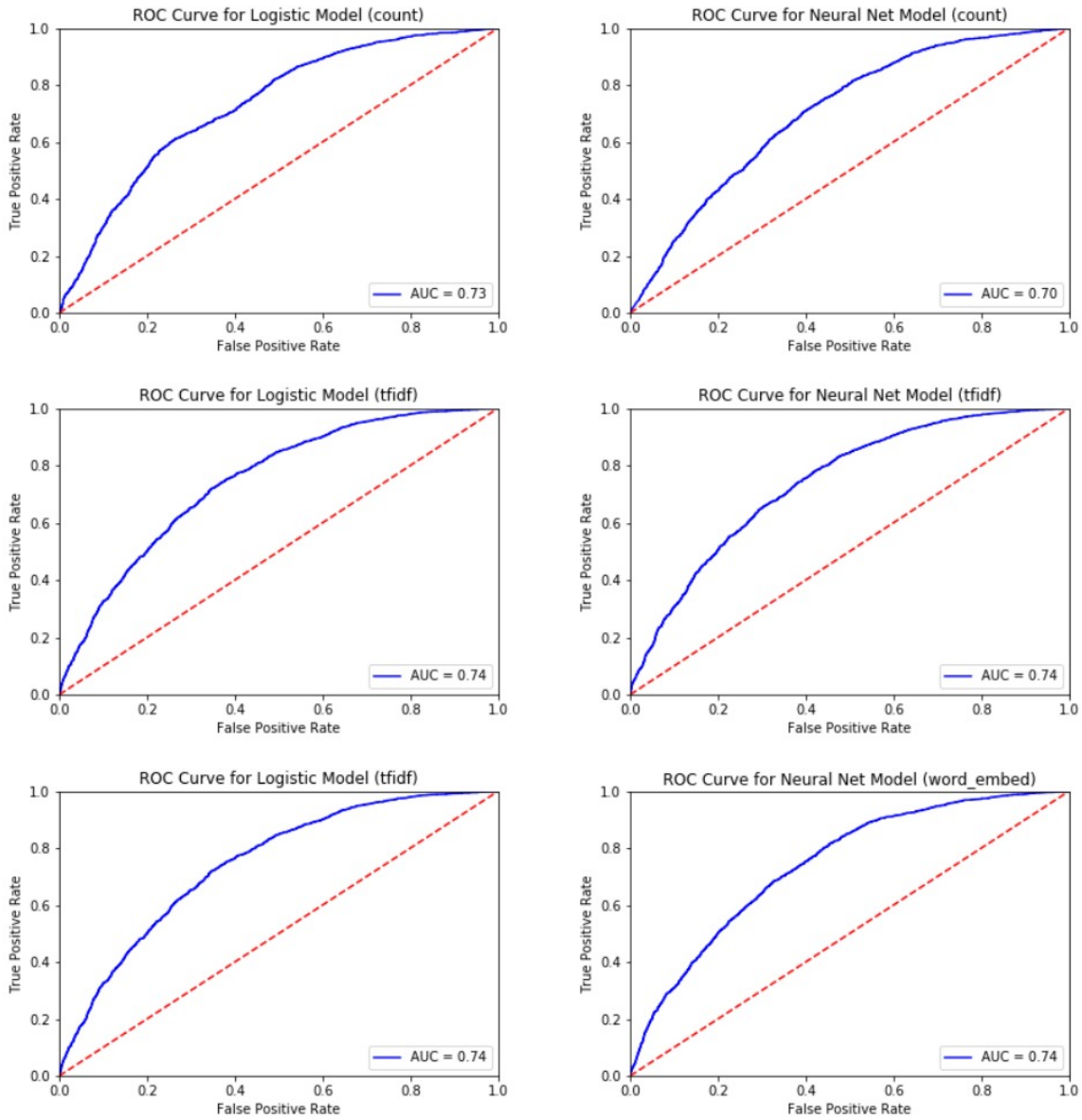

Figure 2: Grid of ROC curves for each of the models. How we embedded the words from each review for the logistic regression and neural network model differed.

learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680. Association for Computational Linguistics.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of Massive Datasets*, 2nd edition. Cambridge University Press, USA.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 43–52, New York, NY, USA. Association for Computing Machinery.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Pro-*

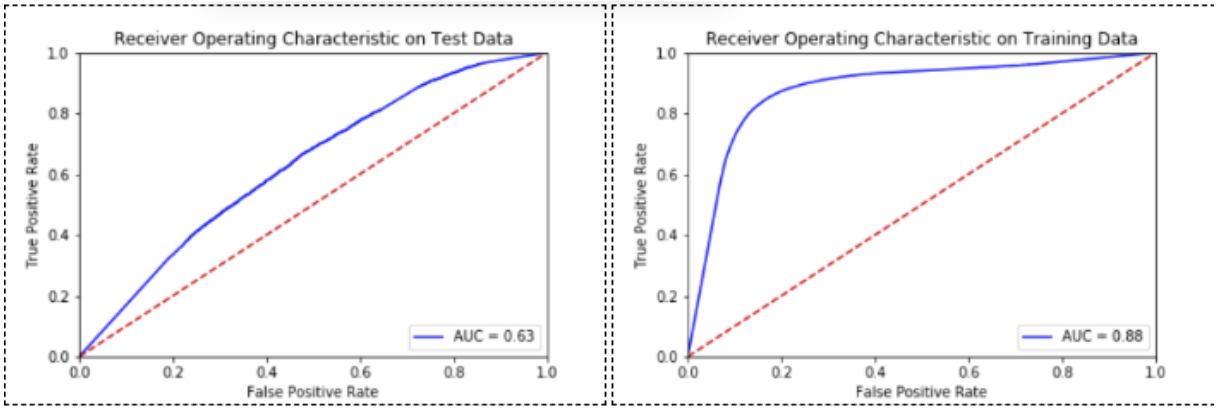

Figure 3: ROC curve of a neural network model on training and test data.

cessing (EMNLP-IJCNLP), pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.