Neural Network for Determining the Helpfulness of Amazon Reviews

Varun Mathur
varun.mathur@epfl.ch

Shikhar Dhar
shikhar.dhar@epfl.ch

Shruti Goli
shruti.goli@epfl.ch

Abstract

The purpose of this project is to predict the helpfulness of Amazon reviews based on the review text. Amazon products generally have hundreds of reviews from users who have previously purchased the product. These reviews explain both the positive and negative experiences that consumers have had with it. From a potential consumer's perspective, these reviews can be overwhelming, confusing, and time-consuming. Determining helpfulness of reviews and ranking them accordingly will give users concise, constructive information when deciding whether or not to purchase a specific product.

1 Introduction

The dataset used for this study comes from UCSD Julian McAuleys "Amazon product data" dataset. We used the 5-core (9.9gb) subset of the data where the items/users have at least 5 reviews. The dataset contains 41.13 million reviews and filters out both users with multiple accounts and plagiarized reviews. Reviews are stored in the form of JSON objects. Each review has the following structure:

```
{
  reviewerID: ID of the reviewer,
  asin: ID of the product,
  reviewerName: name of the reviewer,
  helpfulness: helpfulness rating of
     the review,
  reviewText: text of the review,
  overall: rating of the product,
  summary: summary of the review,
  unixReviewTime: time of the review
     (unix time),
  reviewTime: time of the review (raw
     )
}
```

We used two different models: the first model was built in Keras and the second model was a deep learning model that we made from scratch. The basis for the first model took the reviews and extracted to the vocabulary in the reviews to then predict the helpfulness. The neural network was a single layer model for which you could change the number of hidden nodes in the hidden layer. This model also used a ratio to express words appear more in helpful reviews with words that appear more in unhelpful reviews.

Although the results from both models are not so promising as they only perform, on average, 5%-10% better than random guessing, we have a few ideas on how to improve these numbers. We believe that the dataset is limiting the overall accuracy because it is too broad, thus, running the model on more specific product groups could yield better results. Another idea we came up with is to cluster reviews based on frequency of words. The last idea we had was to use sentence and word embeddings as input for the model. We will discuss these ideas further detail later on in the paper.

2 Background

We believe that there is a direct relationship with the type of words used in the review

and the helpfulness. Thus, our approach to building the helpfulness analysis model was heavily based on current sentiment analysis models. Currently sentiment analysis models are capable of achieving an accuracy above 85% and can run quite efficiently as well by modifying some of the steps in the forward and the back propagation steps.

2.1 Sentiment Analysis Model

The sentiment analysis model is designed to receive data that comes in the format of [[sentence1,sentence2...][label1,label2...]]. Once the model receives the input, it creates a dictionary of the words used in all the sentences and converts each sentence into a binarized list of size equivalent to the length of the vocabulary, this indicates which words are used in the sentence (input layer to the model). From this initial step, you can add more hidden layers to the model to account for more complexities in the data, but for sentiment analysis, one additional hidden layer with an output layer with an activation function, produces results of around 87%.

2.2 Influence from Sentiment Model

We decided on the sentiment analysis model because it is quite a simple model and is effective at determining the sentiment of sentences. In our helpfulness analysis model, we are following the same design structure of the sentiment analysis model, with the input layer representing the words in the sentence, one hidden layer, and one final output layer with a sigmoid activation function ($f(x) = \frac{1}{1+e^{-x}}$) to determine the binary output of whether a review is helpful or not helpful. This is the basic structure of the model we are using; in the following section we will discuss about the modifications we made to the model and how we constructed the labels for each of the reviews to fit a binarized form.

3 Helpfulness Analysis Model

Our approach to creating the helpfulness model was to use a deep learning model. We tried two different approaches when creating the deep learning model, the first was a model built in Keras and the second was a deep learning model written from scratch with tweaks in the forward and back propagation steps to increase the speed of the model.

3.1 Binarizing Labels

Before we could run either of the models, we had to binarize the labels from the helpfulness rating that was given in the dataset. To start, we removed reviews that had 0 ratings, because reviews with no feedback do not provide any information. Next, we binarized the labels by dividing the number of positive reviews by the total number reviews and set a cutoff to decide whether or not a review was helpful or not helpful. The way we decide the cutoff is by looking at the distribution of the labels and taking the average if the distribution is close to normal. Once we had the average we binarized the labels by the cutoff. In the future this step could be improved on so that the average doesn't need to be recomputed every time the dataset changes.

3.2 Model Written with Keras

For the Keras model, we used a densely connected layer with relu activation, a dropout layer, and a densely connected layer with a softmax activation. This model was quite slow and did not prove to be useful as it showed no improvement beyond randomly guessing. Since the model was slow, we decided to write our own deep learning model where we could use optimization to make the training process faster.

3.3 Model

This model is written from scratch in python and is based on common models used in sentiment analysis.

3.3.1 Overall Design

This model has three layers, the first is an input layer, which is the length of the vocabulary being used. This layer reflects the distribution of words in the reviews. The next

layer is a hidden layer, with an adjustable number of nodes, and the last layer is connected layer with a sigmoid activation function, which is used to decide if a review is helpful or not helpful based on a cutoff of .5.

3.3.2 Preprocessing Steps

In this the model written from scratch, we added more prepocessing steps so that the size of the vocabulary would be smaller and contain less filler words, included a check to use only the real words from a sentence, and included a polarity cutoff functionality to the code so that we could change the vocabulary to include more polarizing words. To implement these changes, we first included a way to see what words were used more in helpful reviews and vise versa. We did this by first counting the number of instance a words appeared in either helpful or not helpful reviews. Then we divided by the total number reviews and took the log of that to show that the larger the value the more often the word was used in helpful reviews and the more negative the more it was used in not helpful reviews. From this we were able to remove common words by adding a polarity cutoff. The polarity cutoff would allow us to control the polarity of the words we decided to use while also removing common words. We then added a check to remove not real words from the vocabulary that we were building in the model. This feature can be switched on or off for testing purposes. One reason for not wanting to include this step because there might be some non words that play a key role in determining the helpfulness of reviews.

3.3.3 Optimizations

The optimization we included in our model allowed it to run faster and allowed us to try more number of combinations with regards to the parameters in our model. The optimization in our model was in the training step when the weights were being updated between layers in the forward and backward propagation steps. We eliminated unnecessary steps by adding the weights to update the hidden layer in the forward pass step, and only updated the weights that were used in the forward pass step, so we do not have to do a lot of unnecessary arithmetic in the first step.

4 Findings

4.1 Experiments

With regards to the experiments, we tried many different parameters, different datasets, and different preprocessing methods. Some of the parameters that we changed included the polarity cutoff, the number of hidden nodes, and the the learning rate. We changed the polarity cutoff to see the influence more polarizing words would have on the model. We changed the number of hidden nodes to see the how small we could make the model and still see better than randomly guessing results. Lastly we changed the learning rate to ensure that we were not underfitting or overfitting the model. We also changed the datasets to the see what the results would be on larger and more diverse datasets when compared to smaller, more concise models. Lastly we changed the preprocessing methods to see what changing the words in the dictionary would do to the results.

4.2 Results

The results from the experiments show that the as the size of the datasets increase and the products get more diverse, the results get worse by around 8%. We also found that changing the number of hidden nodes did not do much to change the accuracy of the model. Through experiments, we found that a learning rate of .05 produced the best results. We also found that it was better to not include words that exist in the English dictionary in the preprocessing step. In the following section we will discuss about why we believe our results are consistent with our thoughts.

4.3 Discussion

To begin, we believe that as the dataset size and products get more diverse, the model does worse because the words that the model finds to be good indicators of a helpful review get more diverse and stop explaining what makes a review helpful. In the future, we believe that a dataset with more similar products would produce better results with the same model. We also think that it is better to remove non English dictionary words because they are to specific when it comes to identifying helpful or unhelpful reviews. The model can better generalize the helpful and unhelpful reviews with only english dictionary words.

5 Future Work

Although our model does not show a large difference from randomly guessing either useful or not useful, we have come up with some ideas for future work in this area. As of right now, it appears that the model is unable to understand the significance of particular words because the categories are to large and therefore the helpfulness factor of keywords in these categories are lost. Working with smaller and more concise datasets, where there are more reviews for similar items (i.e. a dataset that consists of only reviews for Pots and Pans) will likely produce better results. We believe that running the model on a dataset of this nature would increase accuracy, because the model would be able to better understand the significance that particular words have on the helpfulness of the entire review. We implemented this in on a small scale and found that as we used more concise datasets, the Amazon datasets with fewer and more similar products, there is around a 8% increase in accuracy when compared to the results that we get when we run the model on the large datasets that have more number of various products.

Another way to approach this problem with the data given, is to cluster the reviews that are in large categories by the words that are in the reviews. Since the data does not provide any smaller categories for the products, one approach to clustering these products would be to look at the words being used the reviews and create categories based on the intersection of these words. The approach to this segmentation task would be the same as k-means clustering, but with the distance function being representing the intersection between the two vocabularies. The vocabularies would be defined by the words that appear in all the reviews for a particular product. Frequency reduction can also be used to remove common words among all products so that the results of the intersection function in the k-means step would be more helpful while also increasing the speed of the entire task. Once the original dataset is segmented, it would allow us to run the model we currently have on more concise datasets, which we believe would lead to better results. One draw back to this approach would be that for any novel datapoint, we would need reviews to identify what category it is in before we could run the helpfulness model on the reviews.

Lastly, we could try to use the word embeddings or sentence embeddings as the input to the model. This would help the model because using embeddings would allow the model to take into account the hierarchical structure of words and sentences. The embeddings would also allow help the model with understanding the relationship between words, which could possibly lead to better results because the model would be better at understanding the helpfulness of particular words or sentences.

References

Bing Liu 2012. *Sentiment Analysis and Opinion Mining Synthesis Lectures on Human Language Technologies,*