Comparison of Generative and Discriminative Language Models for Sentiment Analysis of Product Reviews

By: Suzy Anil, Kristi Van Meter, Suzanna Thompson

Introduction

In this report we will examine product reviews in order to assess if the review is positive or negative using sentiment analysis. To do this, we will use two models; a generative probabilistic language model and a discriminative neural network. For our generative probabilistic language model we have chosen a Naive Bayes approach and for the Discriminative Neural Network we are using a Bidirectional Recurrent Neural Network. Our goal is to compare the results of our two models with respect to the quality and accuracy of the results, the computational cost and efficiency of each model, and the interpretability of each result.

Background

Sentiment analysis is a natural language processing method wherein a text or document serves as the input and a sentiment is delivered as an output. For our dataset, we have chosen reviews of various products and services as our input texts and sentiment options positive or negative as our output. An ideal application of this project could be to aid a company in understanding what aspects of a product could be improved, or determine if a trial of a new product is a good candidate for continued manufacturing without having to manually comb through reviews. This method would save companies manpower and money by reducing the need for an employee to oversee the reviews and classify the sentiment by hand. Additionally, by creating a standardized and reliable method for binning reviews that can be utilized to gather quantitative insights on such qualitative data companies could conduct statistical analyses on the data to further inform business insights.

Dataset

Our dataset was found on Kaggle and contained 50,000 combined reviews from mobile product reviews, Twitter sentiment, Yelp reviews, and Toxic reviews to cover multiple domains of

consumer feedback[1]. The structure of our dataset was such that there were three columns. The first column contained the sentiment, which has the option of positive, negative, or neutral, the second column contained the label of the sentiment, which was coded as 0 for negative, 1 for neutral, and 2 for positive, and the third column contained the original text. During our exploratory data analysis and preprocessing, we found that there was an uneven distribution of sentiments– positive sentiments accounted for 52.36% of the data, neutral sentiment for 19.31% of the data, and negative sentiments for 28.33% of the data. In order to resolve this issue, we omitted the neutral category for sentiment and randomly deleted reviews labeled as positive so that the positive and negative reviews had an equal distribution of 50% each. Our resulting dataset contained 28,330 entries.

Model

Generative Language Model

We chose a Multinomial Naive Bayes Classification Model (MNB) for our generative language model supplemented with TF-IDF encoding, Laplace smoothing, and no prior probabilities of classes. We believe that an MNB is an appropriate model for this analysis because it is a standard classification method of sentiment analysis. Our MNB model should find the probabilities of classes (positive or negative sentiment) assigned to texts by using the joint probabilities of words and classes. Because our outcome variable is binary (i.e., positive or negative), we considered a Binomial Naive Bayes Classifier. Upon running our models, our MNB model produced a higher accuracy score, so we decided against the Binomial Naive Bayes Classifier in favor of MNB. We believe that the accuracy was higher with the MNB compared to the Binomial Naive Bayes Classifier because we used TF-IDF encoding in our models, and MNB tends to perform better when term frequencies are important[2].

Both a Bag-of-Words model and TF-IDF were tested as potential methods of encoding in our model. An advantage of Bag-of-Words was ease of implementation and, given that, low cost of resources for implementation. Because our goal was to provide a method for sentiment analysis

[1] Anshu. “Generic Sentiment: Multidomain Sentiment Dataset.” Kaggle, March 9, 2021.

[2] “Multinomial Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2023.” upGrad blog, November 22, 2022.

to a business, maintaining low resource costs would be important. However, we also tested TF-IDF encoding because we predicted that the element of inverse document frequency would be important in sentiment analysis to capture expressions of satisfaction or dissatisfaction communicated by consumers in more than one word, particularly a pair of words where the first word negates the second (i.e., classifying “not happy” as negative). As predicted, our accuracy was higher for TF-IDF encoded models compared to Bag-of-Words encoded models. Due to that and compatibility with MNB, TF-IDF encoding with MNB modeling was chosen for generative modeling in our analysis.

Discriminative Neural Network

Since the reviews used for this analysis are typically longer text excerpts (>100 words), we needed to incorporate memory into the model so that our predictions utilized both past and future data to output an overall sentiment. Rather than using a Recurrent Neural Network, which uses feedback loops to process sequences in data so memory is involved in the model, we chose a long short term memory network (LSTM). Given the length of the data, we wanted to avoid any loss of information in the network which LSTM accounts for by remembering the significant inputs needed to generate an accurate prediction of the sequence’s sentiment. When reading through some of the training data, there were reviews that started off with positive sentiment but ended with extremely negative words. To take into account how the text might be random in the scattering of its sentimental words, we implemented bidirectional layers in our model to preserve information from the start and end.

First, we needed to define an embedding layer which converts the words into word vectors. Rather than one-hot encoding each word, we wanted the vector to find similarities between the words and for that hyper-parameter we would specify the number of dimensions for the vector as 64. We wanted this neural network to be bidirectional so the input flows in from both directions to maintain the sentiment from previous and future words in the sequence. The layer after embedding was Bidirectional with 64 cells then another bidirectional layer to allow for inputs to be processed in the opposite direction. Then we needed two Dense layers to process the output from the previous cells and convert to the binary classification labels. The first Dense layer included a ReLU activation function with 64 units then the last layer (output layer) contained a sigmoid activation to differentiate between positive or negative labels. In order to

prevent overfitting of the model, we incorporated Dropouts throughout the layers to make sure that noise was not also being learned by the model. Dropout randomly drops a percentage of neurons when moving from a hidden layer, since the neurons are dropped randomly we ensure our model is generalizable to future data. We chose to drop throughout the layers of the model rather than all at once, in order to prevent losing too many important features before calculating probabilities for the classes.

When fitting our model to our training data, we specified values for hyperparameters: epochs and batch_size. We used 64 as our batch_size, so out of training data of size 40,000, each batch would include 64 rows of data which accounts for 625 total batches. The model weights are adjusted after each batch. For epochs, we specified 10, which is the number of times the model will be exposed to the entire dataset or all 625 batches. After each batch, the predictions are compared to the actual labels and the model parameters are updated with gradient descent to further improve the model.

Results

Generative Language Model Results

Interpretation of Real Data Results

When we fit our model with our reviews data, we found an accuracy of 87.45% and our model took about 1.369 seconds to run. This means that our model correctly predicts the sentiment of 87.45% of the reviews. Our Multinomial Naive Bayes Classifier model has the advantage of being highly interpretable. In that, it is easy to explain in broad terms how the classifier works and we're able to relay the probability of a given review being positive or negative. However, our model does not perform well with extremely rare and fake words, such as "Ryy," or "Kopp." Although, that classification problem would also be apparent if a human were deciding a sentiment value as well. Our model also does not perform well with strings of words where every other word has either a positive or negative probability, like "Excellently bad!" Another issue that we did not see but could be a potential blind spot is where the context is not able to be derived from the text. For example, "It is really a budget phone." This comment could be meant positively; as in, "You're really getting a bang for your buck with this phone!" Or, it could be meant negatively; as in, "This phone is cheap and sucks, talk about a budget phone." Again, even

a human would not be able to label the sentiment and would have some personal implicit bias when classifying.

On the other hand, our model works extremely well with complex sentences using Internet Slang, given that there are at least some key markers of the sentiment. As an example, “Fingerprint is working very nice, battery is good with 18 w fast charger, processor is good, Liked everything except the camera, camera quality could be more good.” The model is able to correctly classify this sentence despite it being grammatically incorrect, and using ‘w’ to stand for ‘with.’ On the whole, we believe the model to be adequate as it runs efficiently and quickly without using a lot of memory and has a good accuracy, despite the potential drawbacks.

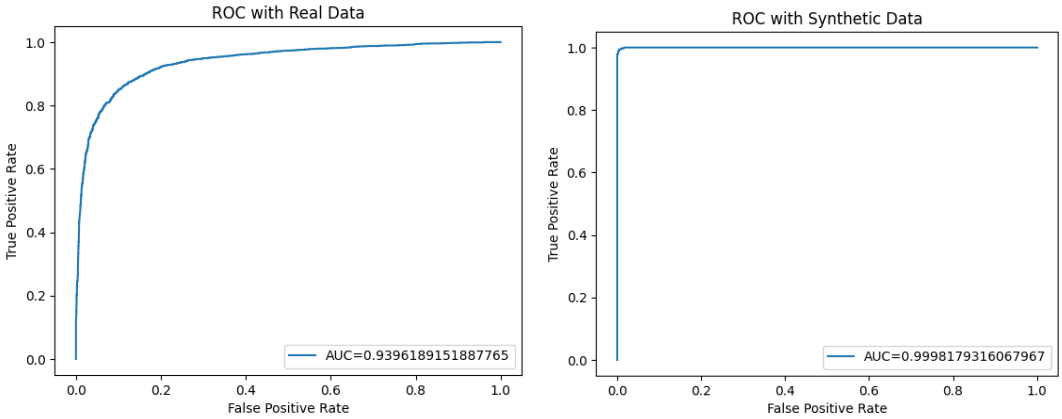

Figure 1.1: On the left, the ROCcurve showing the true positive rate (the rate at which the model correctly labeled the sentiment as positive) against the false positive rate (the rate at which the model incorrectly labeled the sentiment as positive with the real data. On the right, the ROC curve showing the true positive rate against the false positive rate with the synthetic data.

Interpretation of Synthetic Data Results

After fitting our Multinomial Naive Bayes Classifier model with our synthetic data, we have an accuracy of 99.36%, which is considered very good. Fitting the model with the synthetic data took 9.363 seconds to run- about 9x as long when compared to our real data. We believe the increase of computational time can be attributed to how long the synthetic text reviews were. We set a max length of review to be 500 words. Each synthetically created review was assigned a random length, so it is possible that we had a higher frequency of longer reviews that inherently would take longer to process. We should be careful about deciding that our classifier model works well with this synthetic data as it was generated using the classifier model we trained with the real data. In that, the synthetic is generated with a sentiment in mind and picks the next word

by assessing what has the highest probability of being that sentiment. Another point to note is that our synthetic data largely looks like nonsense. For example, we generated the following review: “one nnnnoooooo mi very soaps display 5000mah maxx style trouble scanner.” The model is able to correctly classify this review since the word “trouble” is present, however this review being correctly classified largely has no real impact for deciding the model’s performance as it is very unlikely to see a sentence like that in a real review. So, despite the model working extremely well after being trained on synthetic, the model itself would probably not work very well if tested with additional real data.

Discriminative Model Results

Interpretation of Real Data Results

When fitting our bidirectional LSTM network, we obtained an accuracy of 87.66%. The model took approximately 500 seconds (8.33 min), making it more computationally expensive than the generative model both in time and space. By looking at the results exclusively it is hard to say exactly why this model is misclassifying certain pieces of text, however, we can speculate. Typically, the model has difficulty telling the difference between positive and negative reviews when there is a mix of both positive and negative sentiment words in the sequence. This dilemma was the reason behind creating a bidirectional network over a unidirectional one. However, certain texts have extremely confusing contexts that are even difficult for a human to decipher.

Interpretation of Synthetic Data Results

After fitting our model with the synthetic data that was generated by our Naive Bayes generative model, we observed an accuracy rate of 98.72%. The fitting of the model took 750 sec (12.5 min), which is still more computationally expensive and time-consuming than the generative model. The model did sufficiently well and we even accounted for overfitting by adding Dropout layers throughout our model. Once again, it is difficult to intuitively ration what causes these texts to be misclassified by the model but given that our data was generated by another model, most of the data was incoherent. We implemented a bidirectional LSTM to incorporate the context of the sequence into the prediction of the sentiment and it still did really well in classifying negative from positive. Upon further investigation, we found that reviews with a mix of negative and positive words tend to be misclassified.

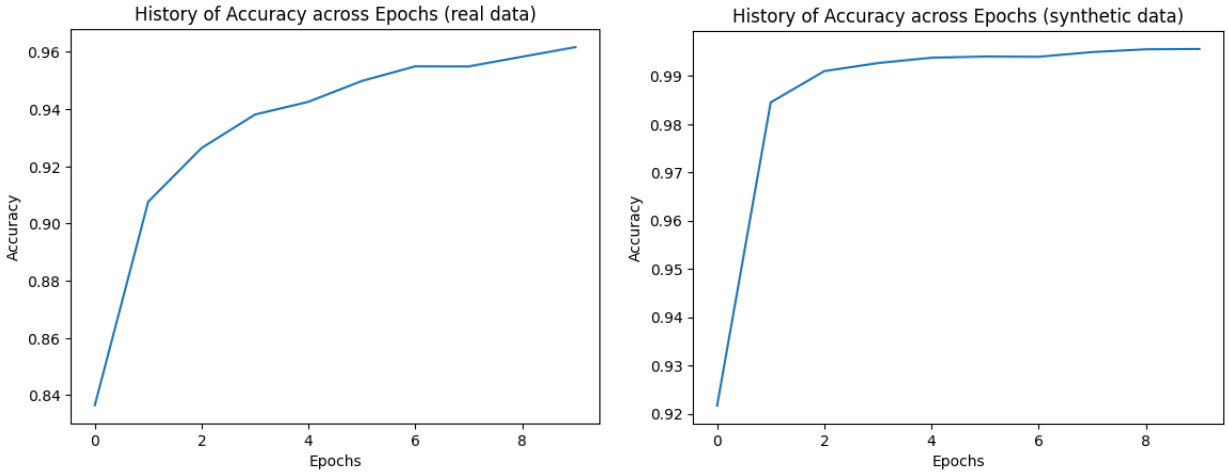

Figure 2.1: On the left, is the trend of accuracy through the ten epochs for the model fit on real data. On the right, is the trend of accuracy through the ten epochs for the model fit on synthetic data generated by our generative model.

Discussion of Methods

We found that the Bidirectional LSTM network had a greater accuracy by 0.21%. However, both models still did well in classifying positive and negative sentiment from both real and synthetic data. The accuracy for both models was highest for the synthetic data and we can attribute this to the fact that the synthetic data was generated with our Multinomial Naive Bayes Classification model, which already had a good accuracy. In the future, it would be of interest to test how well our Multinomial Naive Bayes Classifier model would perform with additional review-based data. We would predict that it would not perform as well as the Multinomial Naive Bayes Classifier model that was originally trained with real data.

The generative model did a great job at fitting itself without using too much computational power compared to the discriminative network, which took about 500x more time to process the same amount of data. While 500 seconds is typically not an infeasible amount of time for training a neural network, this method would not scale if there was a continuous influx of data that required sentiment analysis. The model fitting process for the discriminative model could have been accelerated with the use of additional GPUs or parallel computing, but the generative model would have still outperform the Bidirectional LSTM network, with reference to computational time efficiency.

Another added bonus of the generative model is the ease of interpretability, which can be done by obtaining the probability distribution for each sentiment. This can not be done with the discriminative model, we know gradient descent is being utilized with the learning model but in terms of how and why it is making certain decisions we cannot intuitively assume anything.

A disadvantage of the Multinomial Naive Bayes classifier is that it inherently assumes that all events are independent. For us, that means that we are assuming each review is independent from the others. Our model is limited here as we were not able to conclude if this assumption is met. In that, we're unable to know if each user only submitted one review for a product. So, this method is susceptible to a user 'spamming' a negative review over and over again.

Conclusion

If given extra GPUs and computational power, it is possible that we could have sped up the time it took to train our discriminative model. However, this would have increased the use of capital and our Bidirectional LSTM network would still only perform marginally better than the Multinomial Naive Bayes Classifier model. There is a tradeoff in using either model. With the generative model, we saw better results given that our data was generated by the model itself. We were also able to intuitively understand the results of the model based on the probability distributions. The discriminative network took much longer to fit the model but it was learning and implementing gradient descent iteratively which is ideal when presented with data outside of a specified corpus. For this dataset and specified problem, we conclude that our Multinomial Naive Bayes Classifier model is the optimal model for this problem. However, if we chose to implement this problem at a much larger scale, say within the backend of Amazon's reviews section in order to discern if a product's overall sentiment is positive or negative, we would want to implement the bidirectional LSTM since it can make decisions on its own versus a generative model which can only make decisions based on what it was seen. This, of course, assumes that we would have the computational resources that Amazon could provide.

Citations

Generic Sentiment | Multidomain Sentiment Dataset. (2020). Retrieved December 18, 2022, from https://www.kaggle.com/datasets/akgeni/generic-sentiment-multidomain-sentiment-dataset.

Multinomial naive Bayes explained: Function, Advantages & disadvantages, applications in 2023. upGrad blog. (2022, November 22). Retrieved December 18, 2022, from https://www.upgrad.com/blog/multinomial-naive-bayes-explained/#:~:text=The%20Multinomial%20Naive%20Bayes%20algorithm%20is%20a%20Bayesian%20learning%20approach,tag%20with%20the%20greatest%20chance

Verma, Y. (2021, November 20). *Complete Guide to Bidirectional LSTM (with python codes)*. Analytics India Magazine. Retrieved December 18, 2022, from https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/#:~:text=In%20bidirectional%2C%20our%20input%20flows,future%20and%20the%20past%20information.