Sentiment Analysis of Customer Reviews on a restaurant using Deep Recurrent Neural Network Learning Techniques

Aadhavan Alakan, Domitila Desai, Samarth Lingaraju

Abstract

The sentiment analysis of food reviews has gotten a lot of attention in the recent decade, thanks to the rise of food delivery and cloud kitchen enterprises. The computer study of evaluating people's feelings and views of an entity is known as sentiment analysis. User-generated material through electronic word-of-mouth (WoM) platforms, such as OpenTable and Yelp, is typically deemed more reliable and trustworthy than advertisements from retailers. Our project aims to classify a prominent restaurant's review using deep learning models. The OpenTable reviews were pulled directly from the website using web scraping techniques. The proposed work focuses on classifying the sentiment polarity of user reviews. Traditional machine learning approaches, as well as deep neural networks such as Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM) Networks, and Gated Recurrent Units, were used in the proposed research (GRU). By comparing the results, the proposed system gains a deeper understanding of these algorithms. The suggested work's Deep Learning technique successfully provides a framework for identifying review polarity. This aids restaurateurs in forming a community and promoting a positive image of their companies through regular updates. It also helps them engage with and respond to positive and negative online customer reviews.

1. Introduction

Websites have become prominent venues where consumers share their recommendations and help a business survive or thrive during the last decade. Customer interest and satisfaction are important to the restaurant industry [1]. Therefore, customer reviews on the internet have grown increasingly influential in determining consumer purchases. According to academic data, product information is more important for services than products because services appear to be riskier. As a result, customers who have not personally experienced a restaurant are more prone to seek external information sources [2], [3]. Many internet platforms specializing in restaurant ratings, such as Yelp and OpenTable, provide millions of reviews and attract millions of visitors every day. These companies' success is based on their ability to provide valuable information to users and generate website visitors, which is the key to generating income. From a business standpoint, the concept aids the restaurant's growth by allowing it to learn more about the community and the public's perceptions of the cuisine and service. However, manually analyzing excellent textual reviews is inefficient and time-consuming. As a result, we empower robots to automate the process, which uses artificial intelligence (AI) to detect positive and negative sentiments in text. Sentiment analysis is the term for this procedure. Sentiment analysis is a technique for extracting information from survey replies and product reviews to make data-driven judgments. In this project, our objective is to analyze a prominent restaurant's review from OpenTable; to be specific, we plan to fit a deep RNN model to predict the polarity. Our approach was unique because of its authenticity. We scraped real-time data from the OpenTable website.

We used BeautifulSoup [4] to scrape our data. We scraped around 23,000 customer reviews for our dataset. From many previous works, we know that working with NLP on rule-based traditional machine learning techniques yields an average accuracy, whereas the accuracy standouts when it comes to deep learning techniques. However, deep learning techniques are prone to overfitting. Research shows deeper networks yield better results since they capture non-linearity. Our experiment introduced many deep models like LSTM, GRU, and even combined CNN-LSTM architectures. Our end goal is to build a model that yields good performance without overfitting (low bias and low variance).

First, the customer reviews were unstructured and unlabeled. A lot of prepossessing work had to be done before model fitting. We used TextBlob to predict the processed customer reviews' polarity and labeled all the reviews into three categories Positive, Neutral, and Negative. Then after labeling and preprocessing, we fit the data to various deep models, trained them, tuned the hyperparameters, and finally used them to predict the labels. [5] Finally, We select the best-performing model for testing it with random user-generated data to validate our model. First, we will define a tokenize function that will take care of preprocessing steps, and then we will create a predict function that will give us the final output after parsing the user-provided review.

2. Related work

In sentiment analysis, a given text or comment is analyzed and the prevailing sentiment is captured to determine whether the reviewer's attitude is positive, negative, or neutral. Some of the related works which were found insightful in our project are:

Among similar related works, reviews from sources such as TripAdvisor, Amazon, and IMDB are the text document categories most interested in sentiment analysis in cases of large training data, deep learning methods such as LSTM show better sentiment classification performance with 85% accuracy [7].In sentiment classification the model which helps to classify movie reviews based on the sentiment of sequential text data, it is noted to be effective because it catches up with the classification of long sequence data with the help of LSTM, LSTM utilizes long-term memory, so it can Dealing with long-term dependencies is very effective.[7]

Sentiment Analysis using LSTM, the main advantage of using a CNN layer before the LSTM network helps to identify important features only from the embedding vector, which greatly reduces the training time, making it computationally feasible moreover, the same model can be trained for product reviews and service reviews. No need for complex manual feature engineering, thereby avoiding domain-specific expertise. This is all due to the use of a pre-trained word embedding model to embed the input feature vectors [8] which was referred to for sentimental analysis of tweets, the research team was focused on providing a comparison between sentiment lexicons (W-WSD, SentiWordNet, TextBlob) and validated three of the sentiment analysis lexicons with two machine learning algorithms -SVM and Naive Bayes. In a few cases such as [9] compares Neural network-based sentiment classification methods such as backpropagation, probabilistic neural networks, etc.

3. Data

The proposed method in this work is evaluated on a dataset scraped from OpenTable. OpenTable is a popular online directory for discovering local businesses ranging from bars, restaurants, and cafes. Users can search any restaurants on its website or with the official apps on smart devices. OpenTable has a strong social aspect and often encourages its users to leave written reviews, star ratings, etc. We chose Founding farmer's restaurant for our analysis. It is one of the most popular restaurants in the country, with three franchises near Washington D.C. We scraped all the three restaurants' reviews and combined them for our analysis since all of them were located nearby. The people eating there will be the same population regarding spending power, demographics, etc.

We used the python Requests module [5]to request the website where the reviews are located and then used BeautifulSoup to traverse the result to extract the required parts. BeautifulSoup[4] makes it easier to scan the result and extract data based on patterns from the website. In addition, converting the request result to a BeautifulSoup object (aka making the Soup) made it easier for us to collect the reviews from the users. We then took the raw uncleaned reviews and converted them into a data frame for pre-processing and analyzing the data. Next, we calculated various metrics like words counts, character counts, etc., to analyze the average word length per review.[6]Finally, we calculated the number of Stopword counts per review; Stopwords are commonly occurring words with little to no meaning and do not contribute to the sentiment analysis.

Our next step was to clean the data. This part is critical in every natural language processing, and sentiment analysis as cleaning text differs from regular data cleaning. Well, you're dealing with strings of text rather than data records. [6] Regardless, cleaning the data is still important. The four most common approaches we used were lowercasing all words, removing punctuation, removing [5] Stopwords, and removing trivial words (excessively short and frequent words, e.g., A, I, us). A key part of text analysis is tokenization; this is where blocks of text are split into their words and pieces of punctuation. Performing these steps ensures that the text is as relevant as possible to the message that the actual review is trying to represent. We then use lemmatization. It is transforming your natural text from English to lemming language.

Lemmatization Example:

- (am, are, is) would be lemmatized to (be)
- (car, cars, car's, cars') would be lemmatized to (car)

This reduces the number of available words for analysis by combining similar forms into one base form. We used lemmatization over stemming because we need the content of the review to perform sentiment analysis effectively. We used the Textblob module to lemmatize the reviews. The same module was then used for performing sentiment analysis. The Textblob module again comes in handy for this task and returns not only a sentiment metric but also a subjectivity metric. The polarity metric refers to the degree to which the text analyzed is positive, neutral, or negative, between -1 to 1. A score of 1 means highly positive, a score of 0 means staying neutral, and -1 is considered truly negative. Finally, our dataset is cleaned, labeled, and ready for model fitting.

4. Method

In sentiment analysis of the reviews, we want our model to remember the context and the important keywords which are essential to determine how the customer's feeling was to the experience at the restaurant. To further simplify, having memory in the network is useful because when dealing with sequential data such as text, the meaning of words depends on the context of the previous text.[7]These dependencies have a large impact on the meaning and overall polarity of the sentence or review. In these such cases RNNs particularly LSTMs are being used since they have loops in them which allow the information of previous input or word to persist. LSTMs are capable of learning long-term dependencies; they remember information for long periods of time. This section discusses all the model architectures we used in our project. We used three types of models:

- LSTM (Long Short Term Memory)
- GRU (Gated Recurrent Unit)
- CNN-LSTM (Hybrid model)

4.1 Concept of LSTM model and Architecture

LSTM is a chain-like structure in which there is no single neural network layer, but four gates, which enables LSTM to remove or add information to the cell state (long term).

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous input or word.[8]The equation of forget gate is denoted as ft which comprises of the previous hidden state ht-1(short term), the input of the current state xt and the weights associated with them. A sigmoid function is used to wrap the function and ft is later multiplied with the cell state of the previous input Ct-1. If ft is 0 then the network will forget everything and if the value of ft is 1 it will forget nothing.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Input gate denoted as 'it' is then used to quantify the importance of the new information carried by the input. [9]The new information C^t that is passed to the cell state is a function of a hidden state at the previous state t-1 and input x at present state. The activation function here is tanh, the value of new information will between -1 and 1. If the value is of C^t is negative the information is subtracted from the cell state and if the value is positive the information is added to the cell state at the current state.

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh\left(C_t\right)$$

In the update gate for the present state of Ct it takes information from forget and input gates represented in the [7] equation Ct. The next step is the output and the output gate Ot which is also wrapped around sigmoid function to keep it between 0 and 1. The hidden state of the present state we will use Ot and tanh of the updated cell state. A sigmoid layer is used which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we desire. The LSTM used in this project has the following structure as shown in Figure 1.

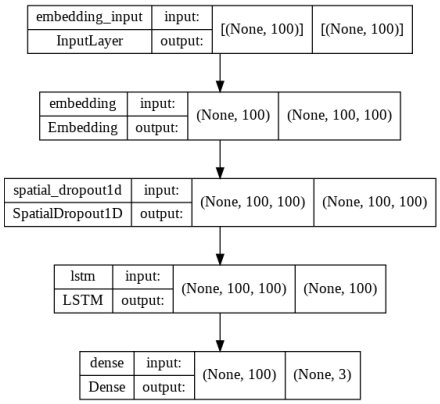

Figure 1: LSTM model architecture

4.2 Concept of GRU model and Architecture

A gated recurrent unit (GRU)[14] is part of a specific recurrent neural network model that intends to use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering, for instance, in speech recognition. Gated recurrent units help adjust neural network input weights to solve the vanishing gradient problem that is a common issue with recurrent neural networks. GRU is a variant of a recurrent network. Like LSTM, it can overcome the gradient vanishing problem of traditional RNN. This is because it introduces an adaptive gate mechanism, which enables the GRU network unit to process current data information and effectively utilize previous data information. This function is completed by resetting gate unit tr and updating gate unit t z together. The forward propagation of the GRU recurrent network is denoted below, in which t x is the input at the moment t.

GRU is a lightweight version of LSTM where it combines long and short-term memory into its hidden state; GRU will have two gates, Update Gate and Reset Gate. The function of the Update gate is that it knows how much past memory to retain, and the Reset gate knows how much past memory to forget. In GRU, the reset gate takes hidden state ht, and the current word and will perform the weighted sum operation and apply sigmoid activation; the rt value obtained is the reset gate value. The update gate performs the same as the reset gate but has different weights; the zt that we obtain is the value of the update gate. Hadamard product is performed, which is a product

between two matrices, and tanh operation is applied for the overall GRU model. The architecture used in this project is depicted below in figure 2.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

Variables

- x_t: input vector
- h_t: output vector
- $\hat{h}_t$: candidate activation vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

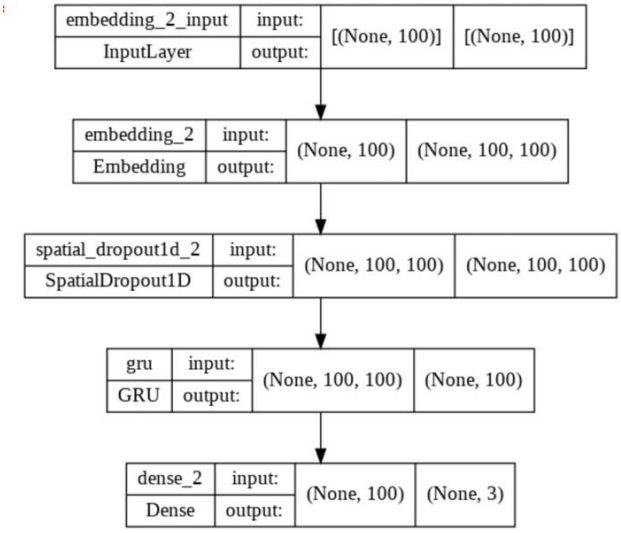

Figure 2: GRU Model architecture

4.3 Concept of Hybrid model and Architecture

In reference to [15] paper for the sequence of words in reviews, the addition of a CNN may be able to pick out invariant features for positive, neutral, and negative sentiment. Furthermore, these learned spatial features may then be learned as sequences by an LSTM layer. Next, the model takes input as word embeddings and feeds them into convolutional layers to extract local features. Finally, the output of the convolutional model is given to an LSTM model to learn the long-term dependencies between the sequence of words. In the end, a classifier layer is applied[16]. The architecture used in this project is depicted below in figure 3.

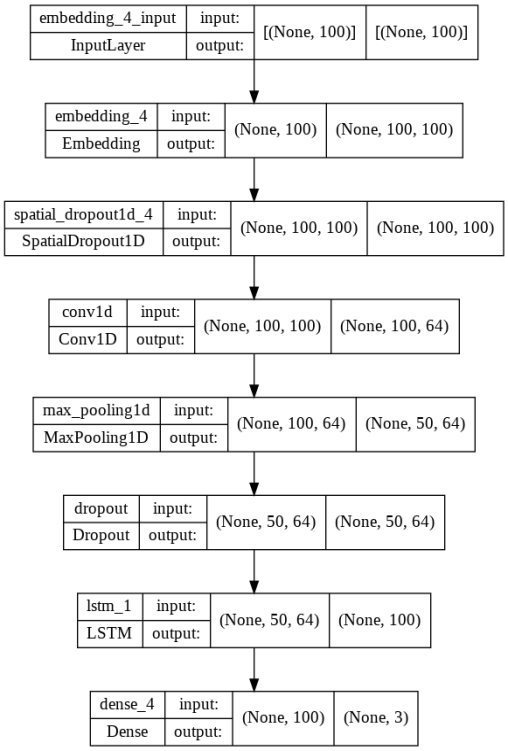

Figure 3: Hybrid model architecture

5. Experiments

In this section, we discuss the parameters we used in our experiments and the results we obtained.

5.1 LSTM

The most important part of model fitting is word embeddings, which are pre-fitted, where words are encoded as real-valued vectors in a high-dimensional space.[10]With the help of Keras, we convert the positive integer representation of words to word embeddings through an embedding layer. We map each word to a real-valued vector of length 100. We also limit the total number of words of interest for modeling to the 50,000 most common words. The sequence length of each review varies, so we limit each review to 100 words, truncate long reviews, and pad shorter reviews with zeros.

The next step was to create a sequential model using keras.models, adding the first layer as an embedding layer to the sequential model with the largest data features, 100 embedding dimensions, and the input data shape. Adding a spatial dropout 1D of 0.4 at this point prevents overfitting by dropping the entire 1D feature map rather than a single element in cases where strong correlations may reduce the learning rate. Next, an LSTM layer was added to the model, which gave 100 units to the dimension of the output space, with hyperparameter dropout and loop dropout of 0.4 to randomly drop neurons in both states to prevent overfitting.[11]The dense layer

adds an activation function "softmax" since this is a multi-class classification setting. The model is then compiled by model.compile(), which configures the model for training. The model is compiled with categorical cross-entropy loss, Adam optimizer for an efficient recommendation, and accuracy.

The results obtained from the model were good; however, we could see signs of slight overfitting. We got an accuracy of 93% in the training data, whereas in the validation set, we got 87%, as shown in figure 4. We stopped after five epochs due to computational complexity plus model overfitting. We assigned 10% of our data for validation. The testing accuracy for our model was 88%, and the F1 score of our model was 88%. F1-score is the harmonic mean of the precision and recall. Figure 5 shows the confusion matrix of the LSTM model.

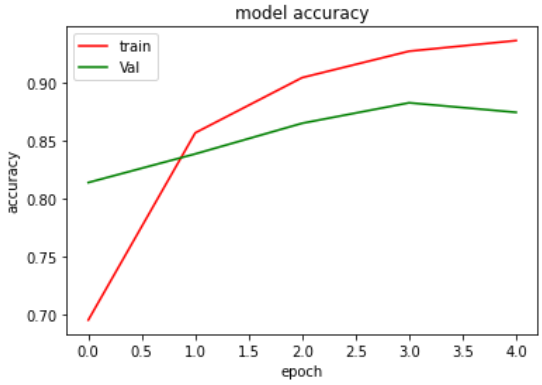

Figure 4: Model Accuracy of Training and Validation dataset of LSTM model

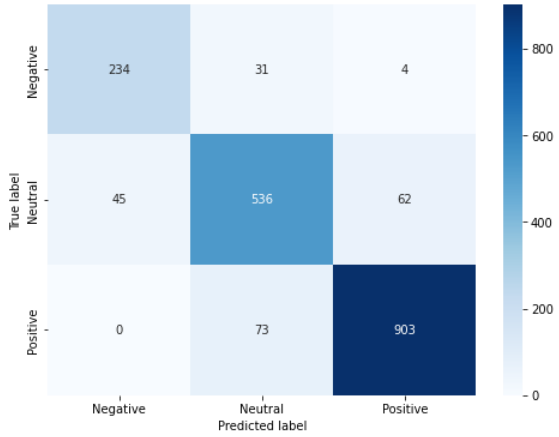

Figure 5: Confusion matrix of LSTM model

5.2 GRU

For our GRU Model[14], we have added word embedding. Word embedding is essential in natural language processing with deep learning. This technique allows the network to learn about the meaning of the words. The first layer of our neural network will perform word embedding, which is essential in NLP with deep learning.[12]These are word representations that allow words with similar meanings to have an equal representation. First, we map each word to a real-valued vector of length 100, just as in the LSTM model, and limit the number of words to 100. Next, we have used special dropout 1D; this feature performs the same function as Dropout; however, it drops entire 1D feature maps instead of individual elements. We have used a special dropout of 0.3 for our model. In the next step, we add the GRU layer by giving 100 units to the dimension of the output space, with a hyperparameter dropout of 0.4 and recurrent Dropout of 0.4, so that it prevents overfitting. Finally, we have added a Dense Layer, Dense implements the operation: output = activation(dot(input, kernel) + bias), where activation is the element-wise activation function passed as the activation argument, the kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer. The output shape of the Dense layer will be affected by the number of neurons/units specified in the Dense layer. We have used a dense layer of 3 in our model.

We have used the activation function as 'Softmax' for our model. Softmax converts a vector of values to a probability distribution. The output vector elements are in the range (0, 1) and sum to 1. Each vector is handled independently—the axis argument sets which input axis the function is applied along. Softmax is often used to activate the last layer of a classification network because the result could be interpreted as a probability distribution. The results obtained from this model were good. We got an accuracy score of 87%, on the testing set and an accuracy score of 93% on the training set, we stopped after 5 epochs as shown in figure 7. The F1 score for our model came out to be 87.6%, and the confusion matrix is as shown in the figure 6.

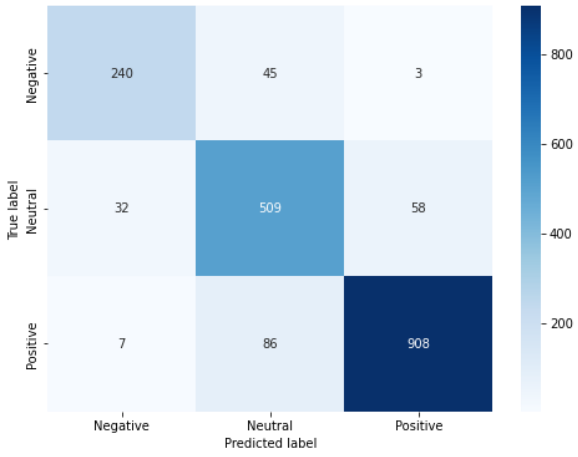

Figure 6: Confusion matrix of GRU model

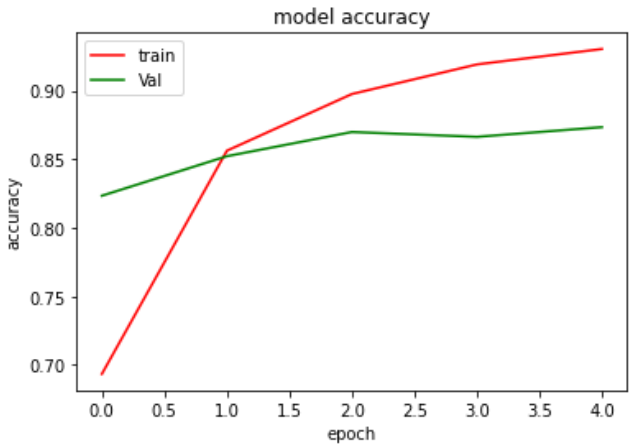

Figure 7: Model Accuracy of Training and Validation dataset of GRU model

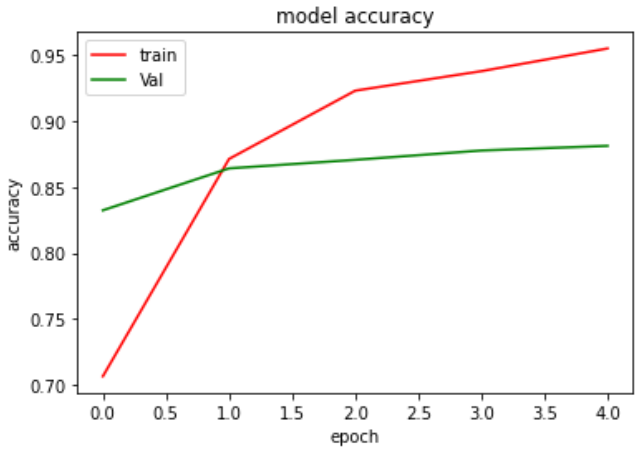

Figure 8: Model Accuracy of Training and Validation dataset of Hybrid model

5.3 CNN-LSTM

We have also experimented with a hybrid model of CNN and LSTM. Referring to this paper [15] [16], we created a sequential model. At the first layer-the embedding, the hyperparameters are the largest data features, 100 embedding dimensions, and the shape of the input data. Additionally, we decided to add a spatial dropout 1D of 0.4 to prevent overfitting since we have limited data and want to minimize the gap between training and validation accuracy so that our model performs best on test data and the misclassification error is minimal. The next layer; is a convolutional layer with a kernel that convolves with the layer input in a single spatial dimension to produce an output tensor. The hyperparameters of this Conv1D layer are 64 filters, the dimensionality of the output space, a kernel size of 5, which specifies the length of the 1D convolution window, and "same" padding that results in uniformly left/right or up /input down so that the output has the same height/width dimensions as the input and a relu activation function to space the output between 0 and 1—adding a MaxPooling layer 1D of size 2 to extract the most important information like feature extraction and dropout of 0.4 to prevent overfitting cases. The output of this convolution is then added as an input to an LSTM layer with 100 units and a dropout of 0.4, resulting in a dense output layer with three outputs with the activation function softmax, which are our three sentiments negative, neutral, and positive of the reviews.

The results obtained from this hybrid model are consistent with the above two models. We got an accuracy score of 88.13%, on the testing set and an accuracy score of 95.49% on the training set, we stopped after 5 epochs as shown in figure 8. The F1 score for our model came out to be 88.08%. Figure 9 shows the confusion matrix of the LSTM model

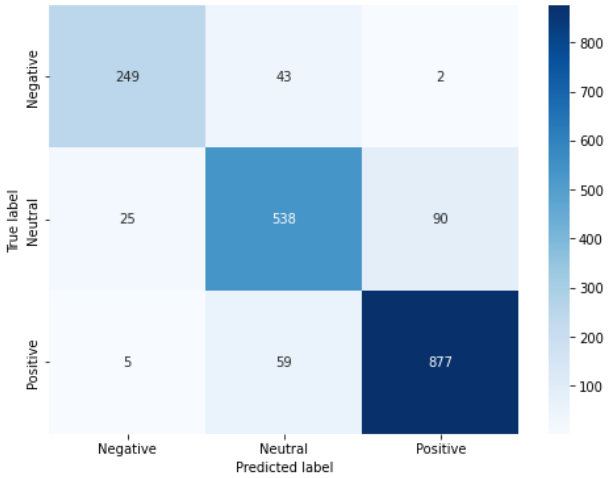

Figure 9: Confusion matrix of Hybrid model

6. Conclusion

The summary of our experiment results can be seen in Table 1 below. We could see that the models' performance doesn't vary much regarding the model accuracy and prediction. This could be because of our access to a limited dataset size. Finally, we selected both the GRU and LSTM models for testing them with user-generated reviews. The model performed well in identifying the polarity of the reviews, thus validating our project.

Model	Dropout	Train Accuracy	Validation Accuracy	Test Accuracy	F1 score
LSTM	0.4	93.66	87.46	88.61	88.62
GRU	0.4	93.05	87.35	87.76	87.85
CNN-LSTM hybrid	0.4	95.49	88.11	88.13	88.08

Table 1: Model Comparison

One of the major challenges we faced is that our dataset consists of only 23485 reviews; the results overfitted the model because of its size. The model's performance didn't improve significantly despite adding regularization parameters like dropout layers (40%); we could only see slight improvements. Hence, we could improve the dataset size so that the model can train well (larger epochs) and predict well on future unseen reviews. In addition, future ideas that could be added to the current model can be an autocorrection function in the preprocessing stage as not all reviews would be correctly spelled. For example, the current model overlooks the misspelled word, which results in substantial information loss if it is a keyword.

References

[1] Ilieska, "Customer satisfaction index-as a base for strategic marketing management",TEM Journal, vol. 2, no. 4, pp. 327, 2013.2.M. Fikri and R. Sarno.

[2] "A Comparative Study of Sentiment Analysis using SVM and SentiWordNet",Indonesian Journal of Electrical Engineering and Computer Science (IJEECS), vol. 13, no. 2019.3.B. Rintyarna, R. Sarno and C. Fatichah,

[3] "Enhancing the performance of sentiment analysis task on product reviews by handling both local and global context",International Journal of Information and Decision Sciences, vol. 11, 2018.

[4]https://medium.datadriveninvestor.com/scraping-movie-reviews-using-beautiful-soup-4-and-python-3-32b0ceeee538

[5]https://www.blog.datahut.co/post/scraping-amazon-reviews-python-scrapy-effectiveness/

[6]https://github.com/Akash321go/Web-scraping-and-text-review-classifiation-using-python-

[7]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[8] “Text based Sentiment Analysis using LSTM”,Dr. G. S. N. Murthy, Shanmukha Rao Allu, Bhargavi Andhavarapu, Mounika Bagadi, Mounika Belusonti

[9] “Sentiment Analysis using LSTM”,Ashok Tholusuri, Manish Anumala, Bhagyaraj Malapolu, G. Jaya Lakshmi

[10]“Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data”, Ranjan Kumar Behera, Monalisa Jena, Santanu Kumar Rath, Sanjay Misra

[11] “Sentimental Analysis of COVID-19 Tweets Using Deep Learning Models”,Nalini Chintalapudi, Gopi Battineni, Francesco Amenta

[12] “A comparative performance evaluation of neural network based approach for sentiment classification of online reviews”, G.Vinodhini,R.M.Chandrasekaran

[13] “Analyzing and Filtering Food Items In Restaurant Reviews: Sentiment Analysis And Web Scraping” Nina Luo1, Caroline Kwan2, Yu Sun3, Fangyan Zhang4

[14] “A GRU Model for Aspect Level Sentiment Analysis”, Yongping Xing and Chuangbai Xiao 2019

[15] “A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents” Prafula Kumar Jain, Indian Institute of Technology (Indian School of Mines) Vijayalakshmi Saravanan, Rochester Institute of Technology Rajendra Pamula, Indian Institute of Technology (Indian School of Mines)

[16]“A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis” Prafula Kumar Jain, Indian Institute of Technology (Indian School of Mines) Vijayalakshmi Saravanan, Rochester Institute of Technology Rajendra Pamula, Indian Institute of Technology (Indian School of Mines)

[17]http://karpathy.github.io/2015/05/21/rnn-