San Jose State University

Data 255 Team Project

Team 4: Kristie Nguyen, Tien Nguyen, Ziwei (Sophia) Song

**Yelp Sentiment Analysis Report**

1. **Project Background & Approach**

Yelp is a well-known company that developed a website and app where users are able to write reviews and rate businesses on a star rating of 1 to 5. By creating this functionality, users are able to see which businesses have good reviews and the businesses are able to see how their customers evaluate their services. However, for many businesses, there are over hundreds of reviews left by their customers and it would be tedious to manually shift through the reviews. This is where Natural Language Processing (NLP) comes in. In this project, we built a text classifier that focuses on restaurant reviews using Python’s Pandas, NLTK, and Scikit-Learn libraries. The objective is to predict whether a review is positive or negative based on the context of the reviews. By implementing a prediction algorithm, we can create a daily report for the restaurants and provide a sentiment analysis of the reviews. Humans express their opinions and emotions differently, so the complexity of the language makes it difficult to accurately analyze the tone and context. Since we are doing sentiment polarity on the reviews, we may encounter several roadblocks regarding the interpretation of the English language. Some of the challenges we might run into are irony and sarcasm, word ambiguity, negation detection, and multipolarity.

1. **Data Exploration and Understanding**

*2.1 Data Collection*

We collected our data from Yelp Open dataset which includes two json files, yelp\_business.json file containing basic business information and uelp\_review.json containing customer reviews. The original size of the dataset is larger than 6GB. To process the data that can be used as training data for our model, we first downsized the dataset with three criterias: a. Drop rows with blank fields; b. Filtered category keywords including “Restaurant”; c. Category label length less than 4. With the data preprocessing, we have a cleaned and normalized CSV file as yelp\_business.csv and the corresponding yelp\_review.csv. Then merge the business information with corresponding reviews by business\_id and create a new csv file with all the columns we need for further analysis.



Figure 1. Overview of merged CSV file

*2.2 EDA for business.csv*

After cleaning and selection, there are 8 columns in the file: business\_id, name, city, state, stars, review\_count, categories and label\_length. To understand each column, besides the first four columns, we will explain the rest features.

1. Stars: overall review stars for each business.
2. Review\_count: the number of reviews for each business.
3. Categories: the category of business, may include multiple categories for each business
4. Label\_lenght: each category is counted as one label, label\_length counts the total number of labels for each business.



Figure 2. Overview of business.csv

To better understand the data, we are interested in exploring if the count of store count is evenly distributed across stars and review count. For each feature, we plotted a barchart.

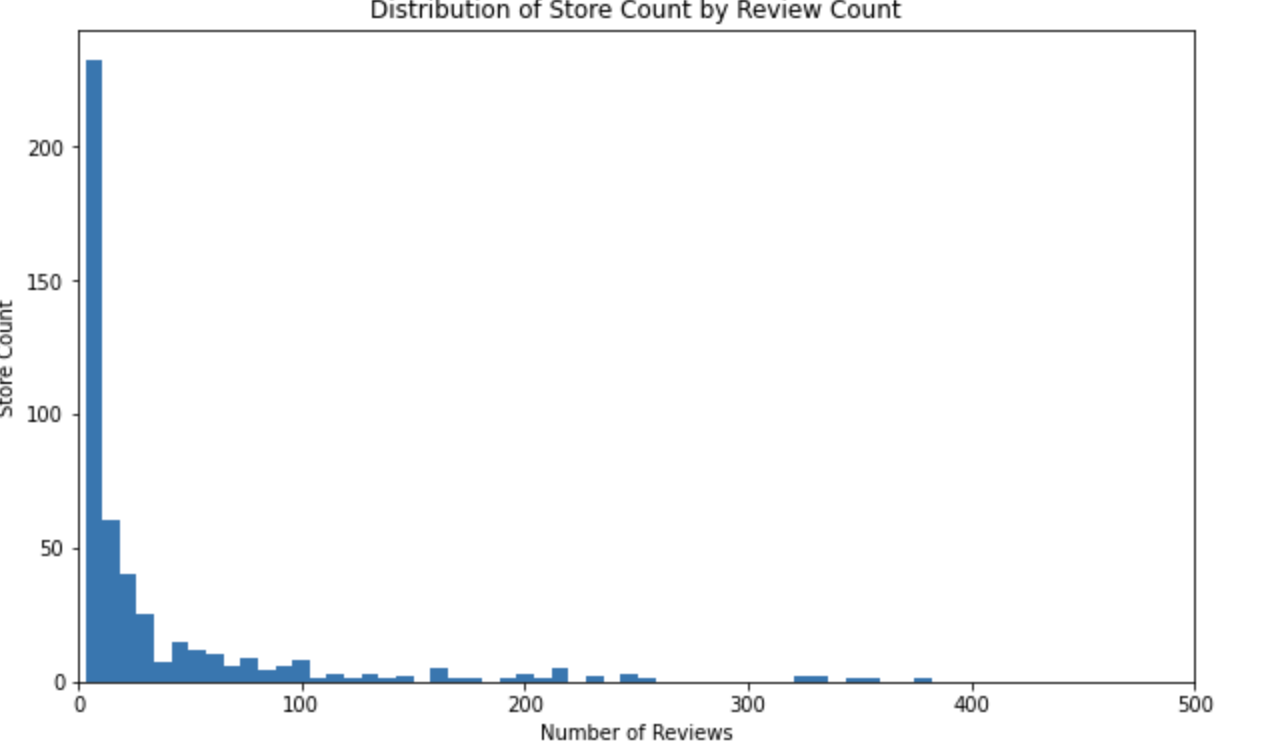
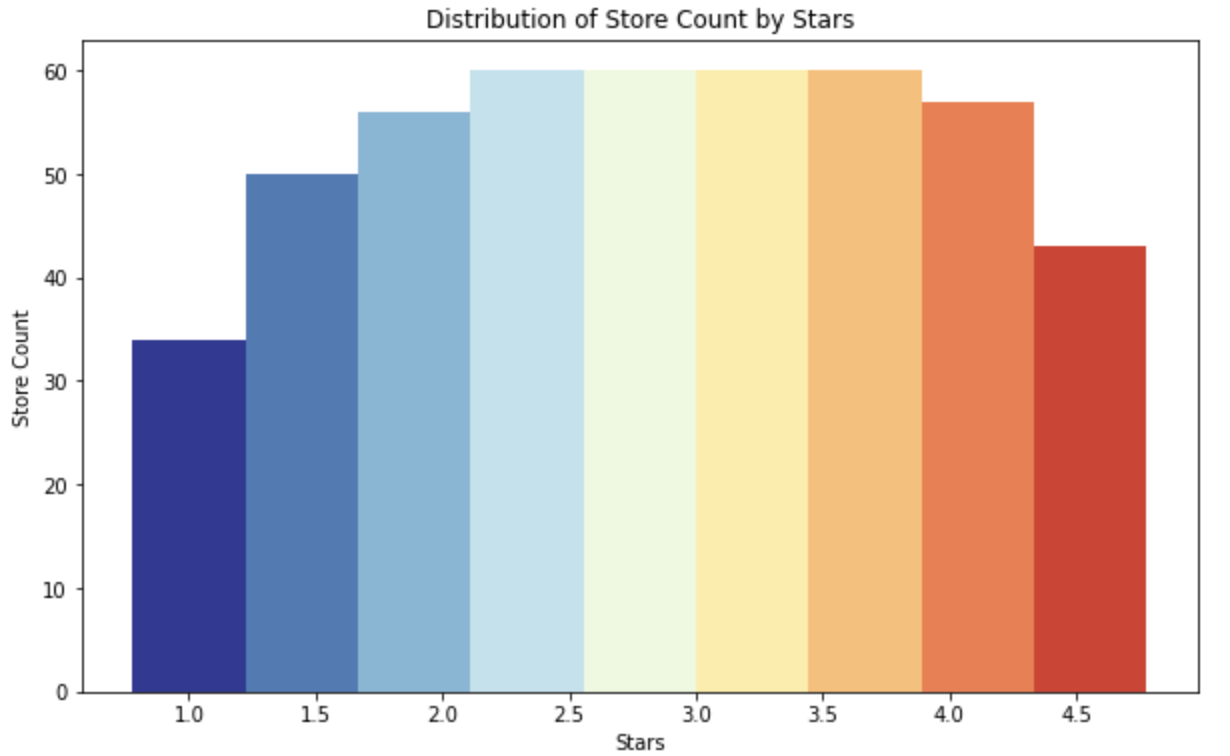


Figure 3. (a) Distribution of store count by Stars, (b) Distribution of store count by review count

From figure 3(a), we can see that the majority of the stores are scored with star 2.0 to 4.0. From figure 3(b), we can tell that most of the businesses have less than 50 customer reviews.

*2.3 EDA for review.csv*

The review.csv file extracts related information from Yelp review section. Figure 4 is an example from the Yelp App review board. Review\_stars refers to the star each reviewer gives to the business and columns useful, funny, cool refer to the other customer’s feedback on the specific review text. Figure 5 is the overview of data which includes all the information mentioned above.



Figure 4. Example of Yelp review

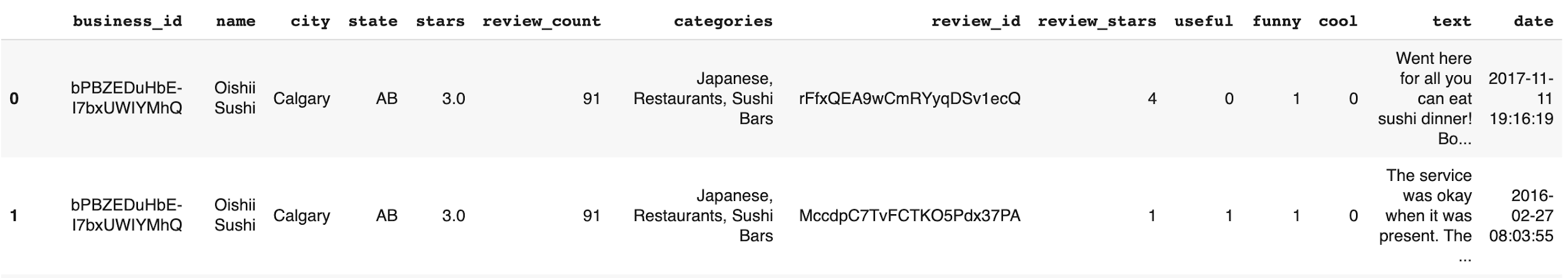


Figure 5. Overview of review.csv

To better understand the data, we explore the relationship between text length and review stars by plotting separate bar charts as figure 6(a). In addition, we use the box plot to show the shape of distribution with central values and variabilities in the figure 6(b).

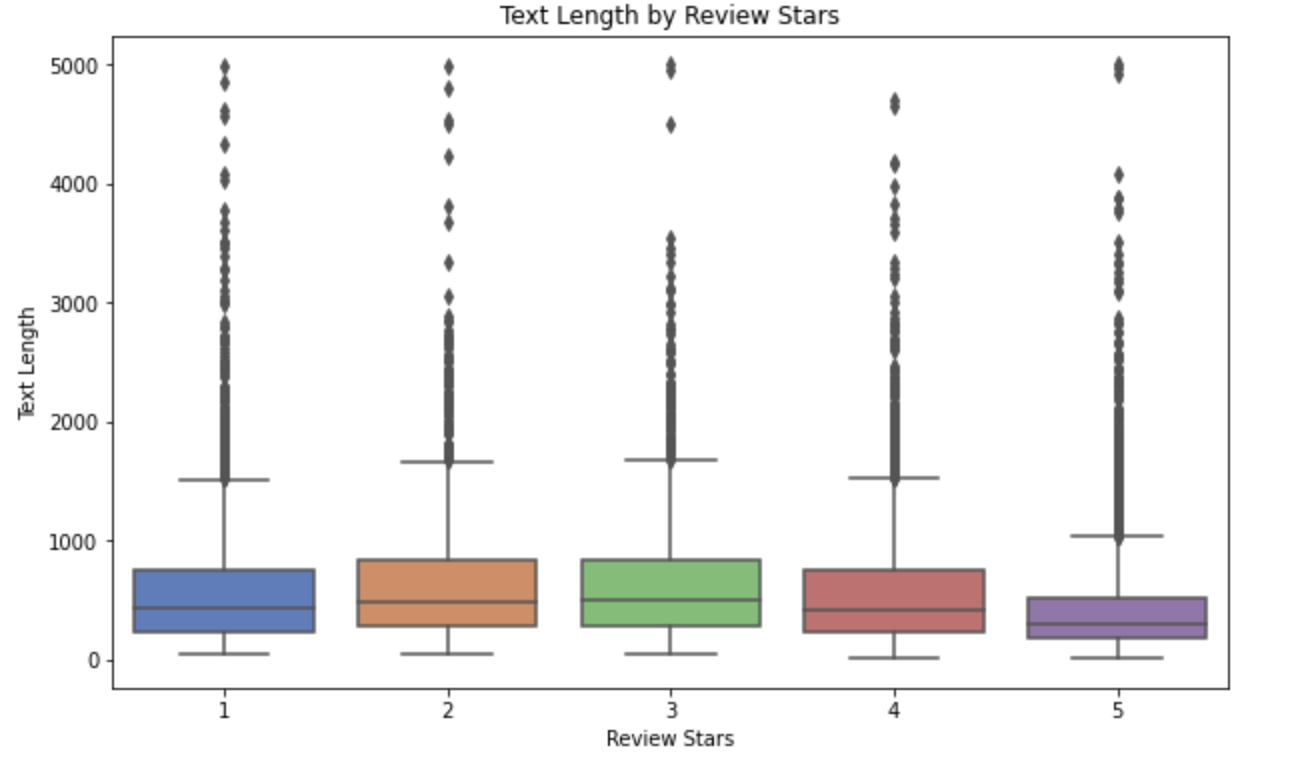
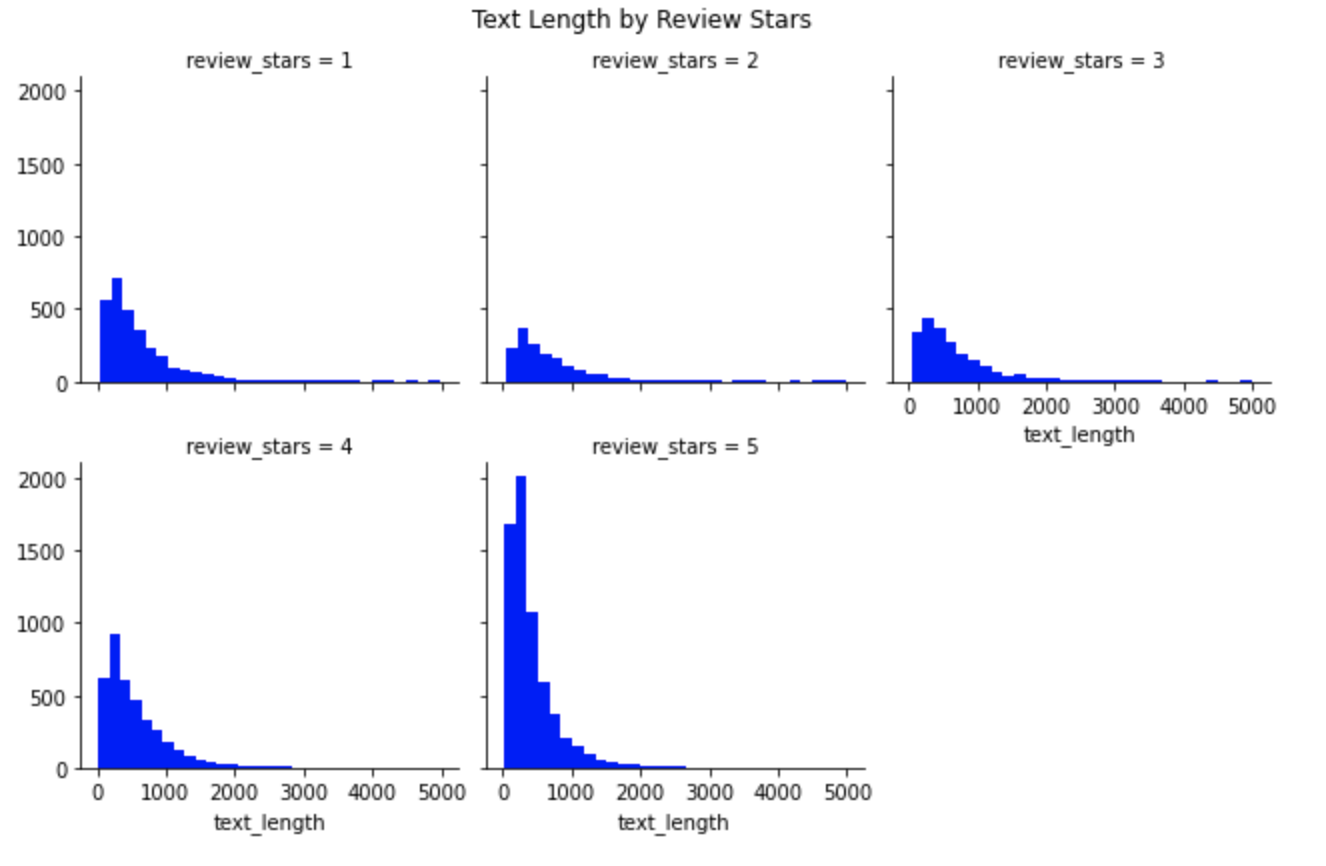


Figure 6. (a) Bar Charts; (b) Box Plot

Besides the text length, we also concern the relationship between the number of reviews with review stars. Figure 7 shows that people who leave a 5 star review tend to write comments for the business.



Figure 7. Review Count by Review Stars

The customer review type is also crucial to explore. We plotted the line plot to display shape of distribution for each review type by review counts and stars. Figure 8 shows that distribution of review type by star is concentrated on 3.5 and 4 star reviews.

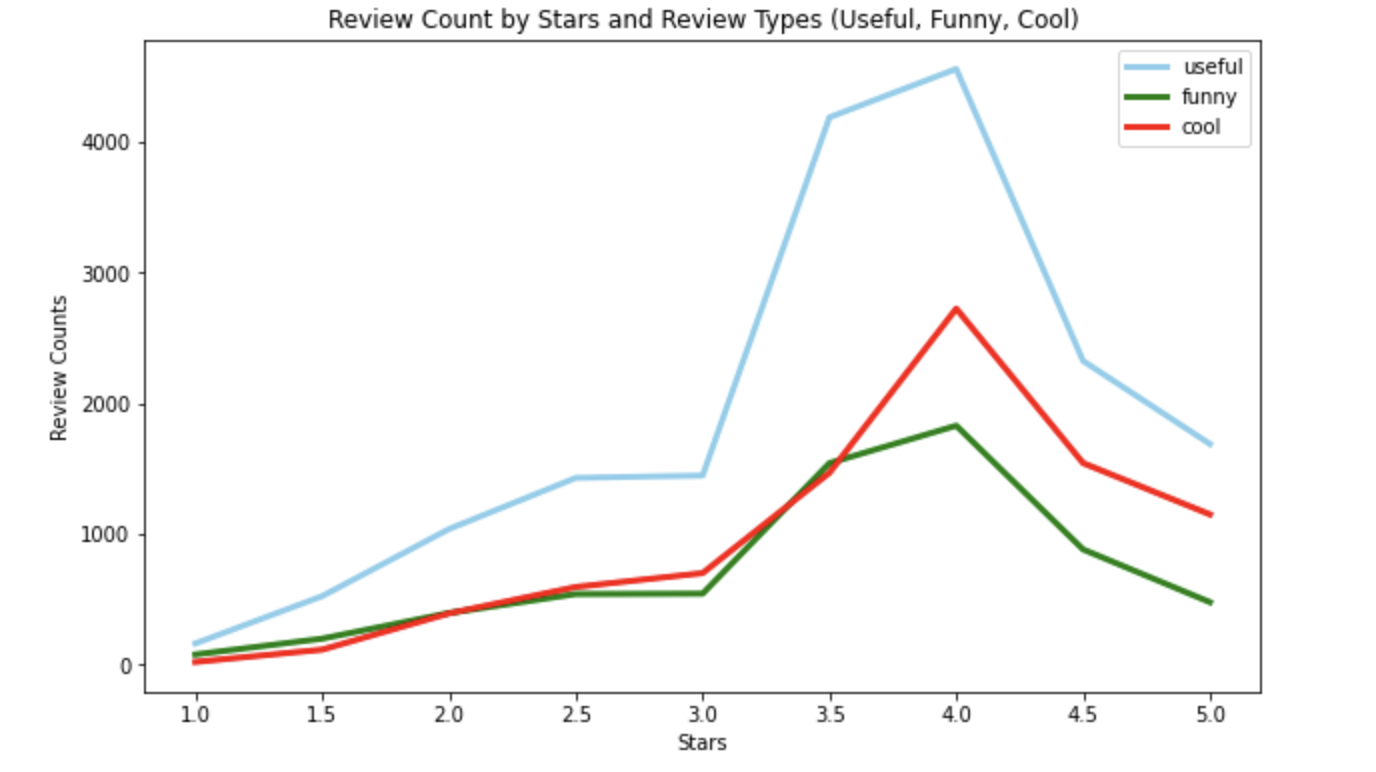


Figure 8. Review Count by Stars and Review Types

1. **Data Preprocessing**

The data preprocessing includes five main tasks: text cleaning, text normalization, tokenizer and pad sequence, prepare train/test data set, and creating word embedded vectors.

*3.1 Text Cleaning*

The input is text string, so we apply the following techniques to clean up the text

* Stopwords removal: ***nltk*** stopword library is used to remove common stopwords in the text string
* Punctuation removal: remove all punctuation in the text string
* Lowercase conversion: convert all character to lowercase
* Number removal: remove numbers in the text string
* Whitespace removal: substitute multiple whitespace with single one, remove leading and trailing whitespaces

*3.2 Text Normalization*

The following methods are used to normalize the text:

* Lemmatization: ***nltk*** library is used to lemmatize similar words into a single form
* Word Stemming: PorterStem from ***nltk*** is used to remove the commoner morphological and inflexional endings from words to create stem of words

*3.3 Tokenizer and Pad Sequence*

We use Tokenizer from keras to split our text string into smaller units, and only keep a maximum of 10,000 features. Pad\_sequences from keras is also used to ensure that all sequences have the same length of 200. The values for max features (10,000) and max length (200) will be changed to improve the model performance in the training and validation process.

*3.4 Train/Test Data Split*

We use train\_test\_split from sklearn to randomly split our data into 70% training + validation and 30% testing.

*3.5 Word Embedded Vectors*

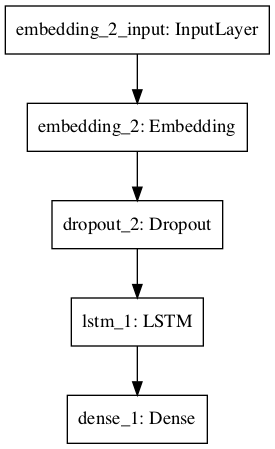
We implement two different approaches to create word embedded vectors:

* Regular embedded layer using keras
* Custom embedded layer using Word2vector

1. **Model Implementation**

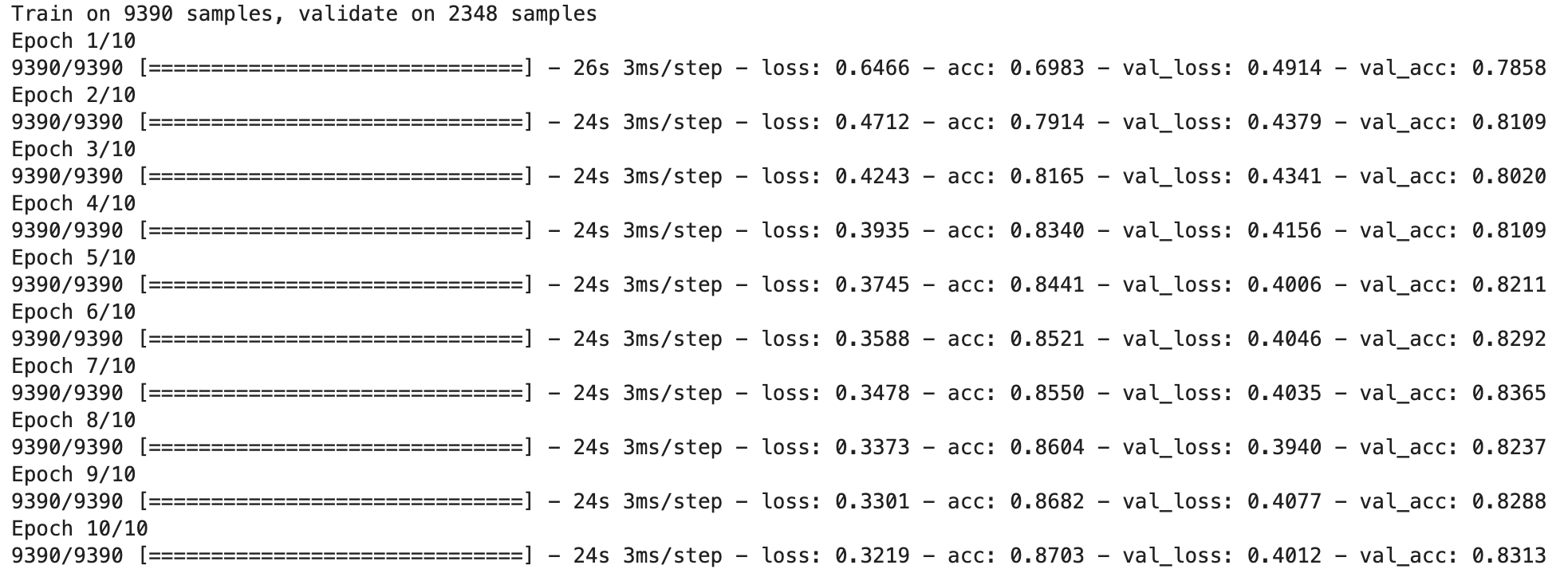
*4.1 RNN*

The first RNN model consisted of an embedding layer, a dropout layer, a LSTM layer with dropout, and the output from LSTM was fed into a hidden fully connected layer. For the embedding layer, we passed 10000 features named as max\_words, input length 200 as maxlen, and 32 as the timestep to form the dimension of the embedding layer. For the LSTM layer, the first parameter is the same as the third parameter from the embedding layer, and we added a 30% chance of dropout with L2 regularization. The activation function used in a dense layer is sigmoid function.

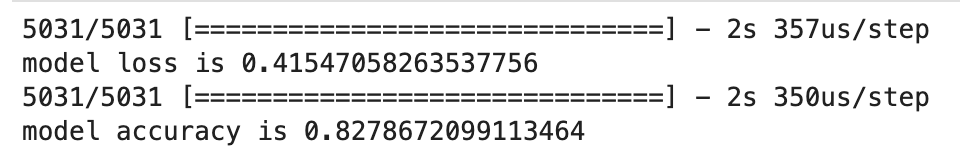


Model Architecture

With the above model architecture, we trained the model with batch size 32 in 10 epochs, and set split\_validation as 20%. The training results are shown below and the best test accuracy is 82.787%



Training Result



Test Accuracy: 82.787%

*4.2 Bi-directional LSTM*

Bi-directional LSTMs are an extension of LSTMs of traditional LSTMs that are used to improve model performance on text classification problems. Unlike traditional LSTMs, bi-directional LSTMs are trained using two LSTMs instead of one on the input sequence. First, the tokenized sentences go through an embedding layer, where it modified the integer representation of words into dense vectors. After the embedding layer, it connects to the dense layer where the softmax and sigmoid activation functions are used.

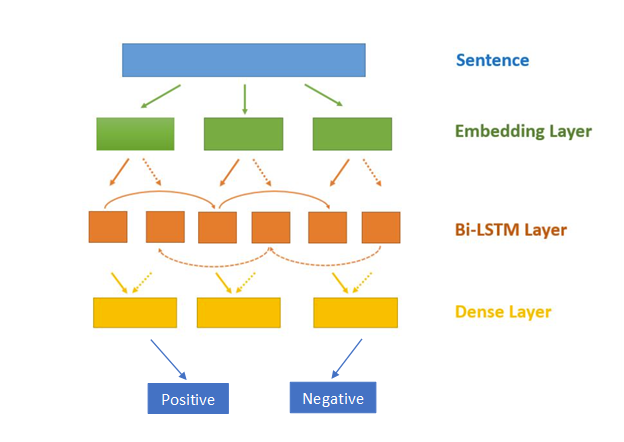
**

Figure 9. Bi-directional LSTM Model Architecture

*4.3 Hybrid Model: CNN + LSTM*

In the effort of improving the text classification accuracy, we also implement a hybrid model: CNN + LSTM. First of all, we create an embedded matrix from the customized embedded word vectors. Next, it is fitted into the hybrid models, which is considered as two sub-models: the CNN model for feature extraction and the LSTM model for interpreting the features across time steps. Figure 10 below shows the architecture of our hybrid model.

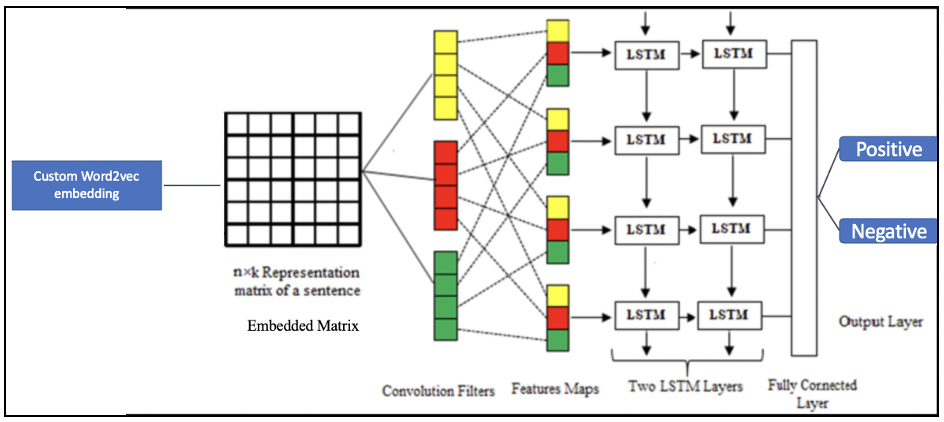


Figure 10. Hybrid Model Architecture

1. **Model Results and Evaluation**

*5.1 Model Results*

5.1.1 RNN

From the loss curve graph on the left, it showed both training and validation loss are decreasing for the first 6 epochs. After the sixth epoth, the validation loss starts increasing to 0.4. Compared with the accuracy graph on the right, the best accuracy result for validation is at the sixth epoch as well. Besides the RNN architecture mentioned in 4.1, we also tested the model on different sets of parameters. Figure 12 shows the results before adding dropout layer and adding dropout rate in LSTM layer. We can conclude that overfitting issue has been reduced for a large content as we adjusted the layer parameters.

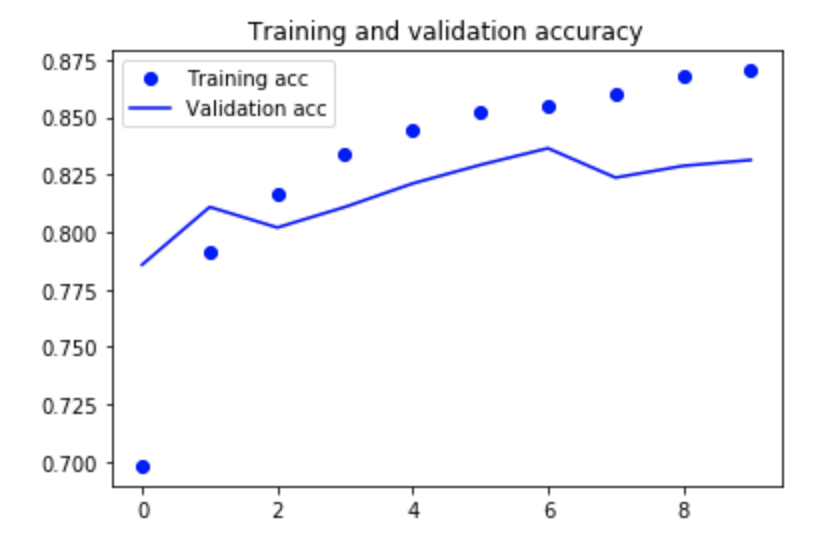
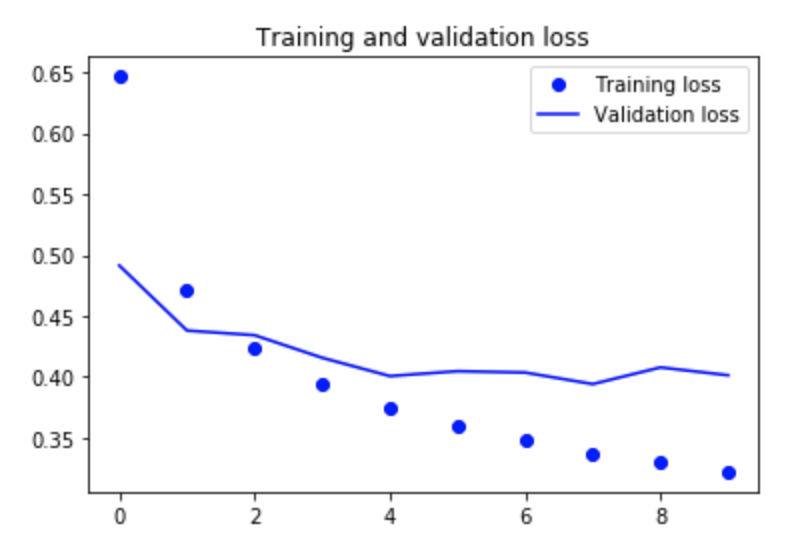


Figure 11. RNN Training/Validation Loss and Accuracy

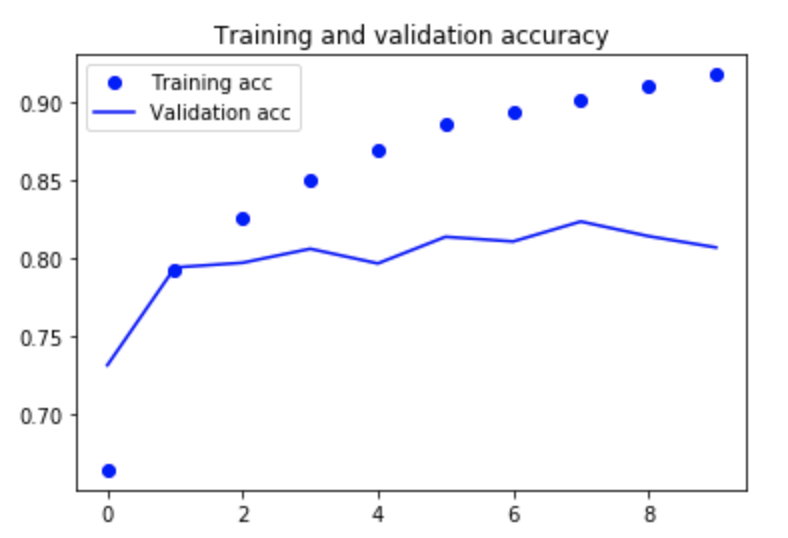
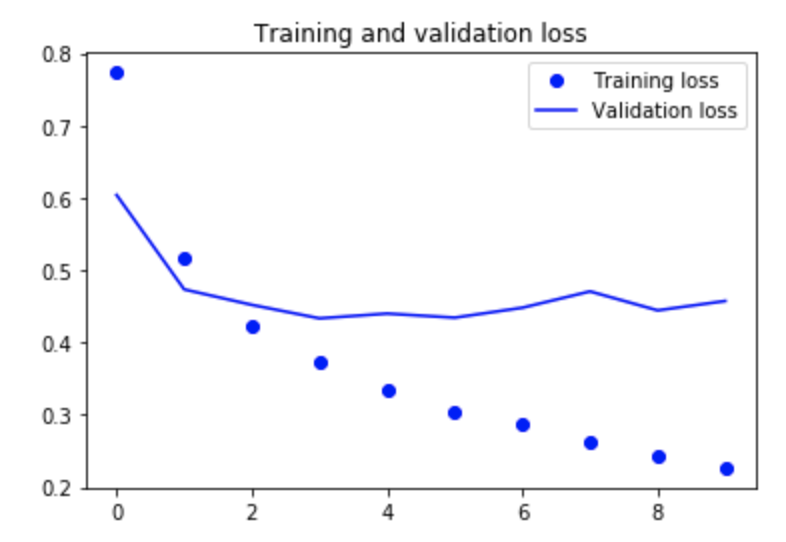


Figure 12. RNN results with overfitting issue

5.1.2 Bi-directional LSTM

As the number of epochs increased, the training and validation loss closed the gap, which showed that there was not an overfitting problem. The accuracy curves also proved that there was not an overfitting problem since the accuracies of the train and validation were relatively in the same range. Adding dropout layers helped the prevention of overfitting, so that’s why this model worked well. With this model, it was able to achieve an accuracy of 82.23%.

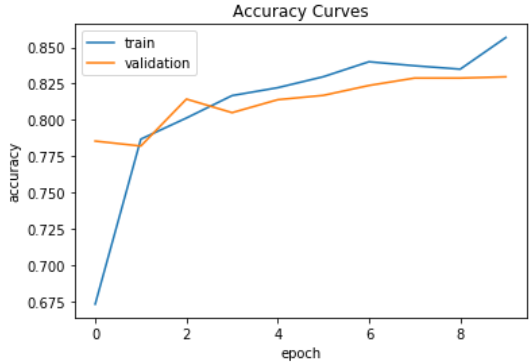
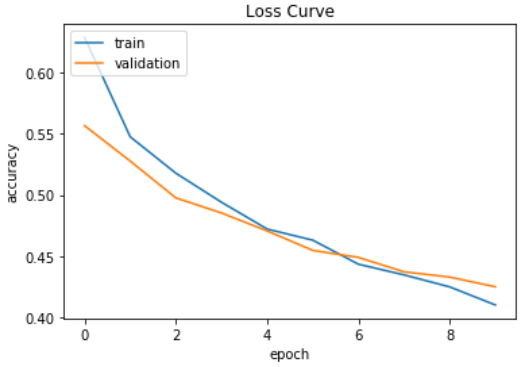


Figure 12. Bi-directional LSTM Loss Result

5.1.3 Hybrid: CNN + LSTM

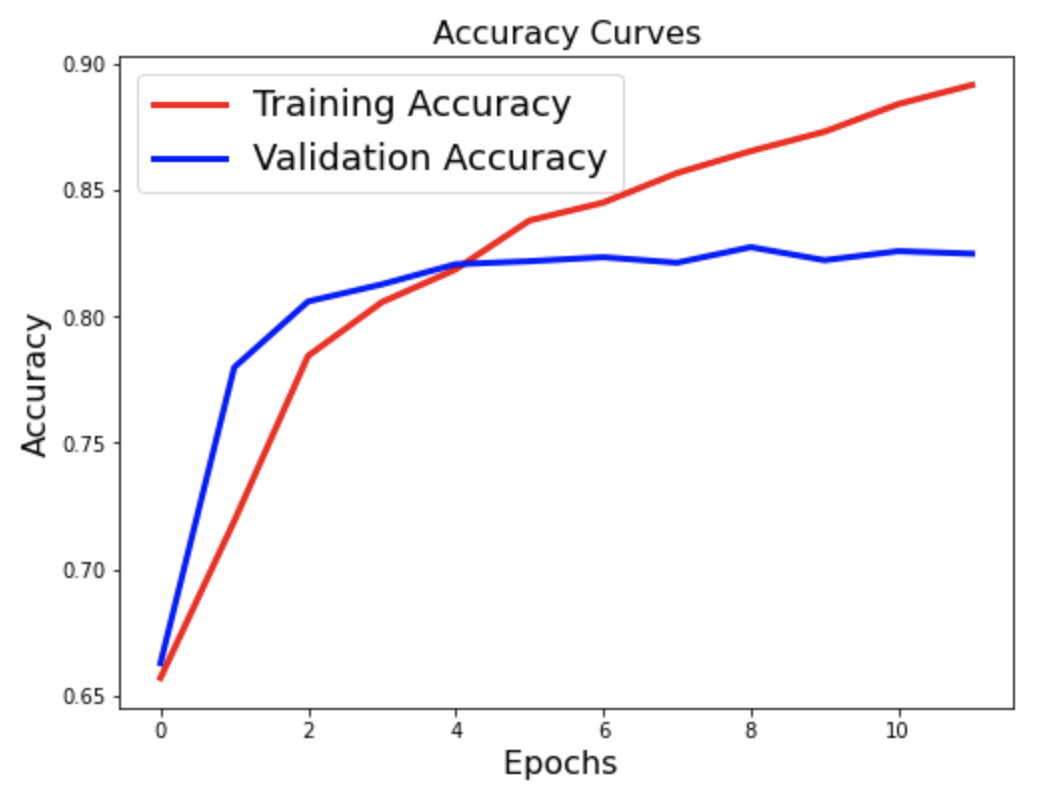
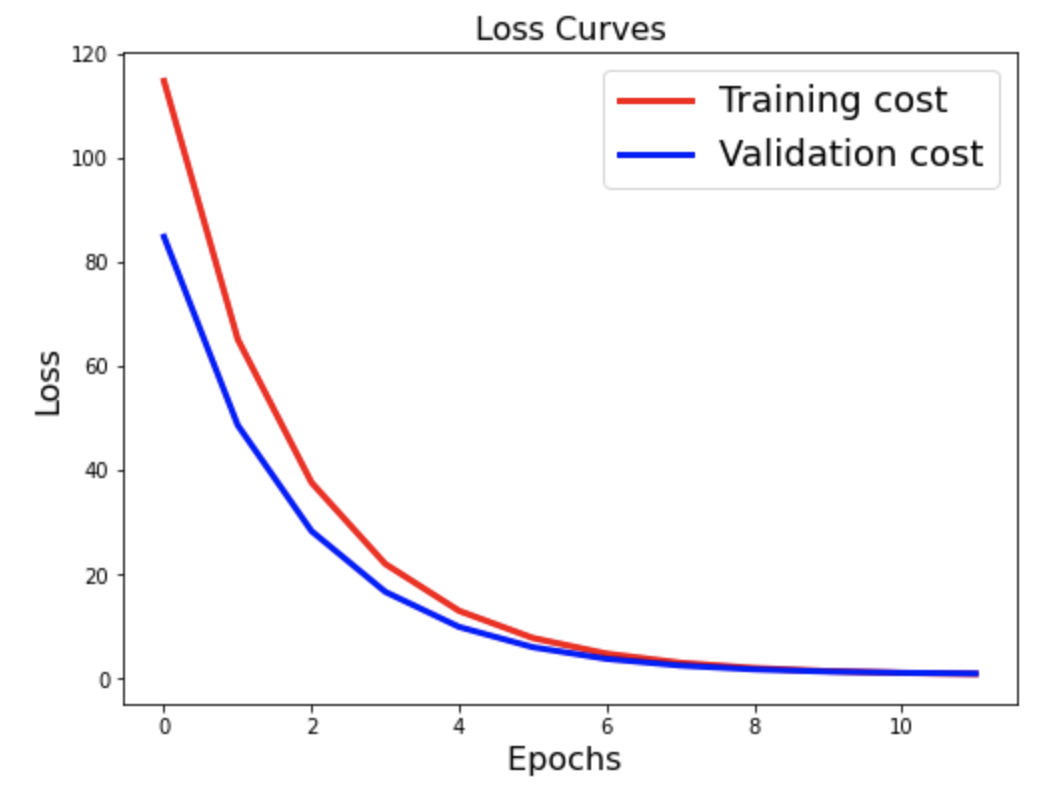


Figure 13. Hybrid Model Training/Validation Loss and Accuracy

Through each epoch, the training and validation loss decreases to a point of stability with a small gap between the train and validate loss lines. On the accuracy, training accuracy keeps increasing but validation ones in creases in first few epochs, but after that, it doesn’t seem to increase anymore. With this hybrid model, we are able to achieve ~82.5% accuracy on the test set.

*5.2 Model Evaluation*

Training a deep neural network that can generalize well to new data is challenging. In this project, our models have not generalized well and have issues with overfitting. The deep neural network models perform well on the training data set but poorly on the test set. We address this overfitting issues with the following methods:

* Dropout: probabilistically remove inputs during training
* Early stopping: monitor model performance (validation accuracy) on the validation set and stop training when performance degrades

To evaluate our model results, we use the following metrics:

* Accuracy: measure how often a sentiment rating was correct
* Precision: measure ratio of correctly predicted positive comments to the total predicted positive comments
* Recall: measure of our model in terms of correctly identifying the True Positives

1. **Model Comparison**

As a result of our models, the accuracy of each model was very close to each other, so it was difficult to point which model performed the best or worst. In Figure 14, there is a bar chart visualization of each model’s accuracy, precision, and recall, which all ranges around 83%-88%. For other traditional methods like Naive Bayes, the accuracy can only reach up to 70%. This shows that deep learning methods are better algorithms for sentiment analysis.

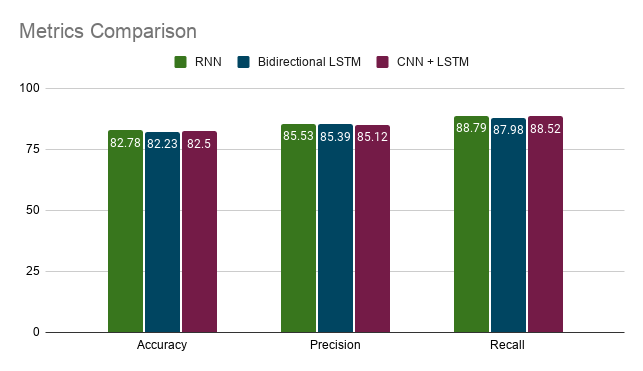


Figure 14. Model Comparison

1. **Future Work & Deployment**

*7.1 Future Work*

In this project, we are able to classify positive/negative comments with the accuracy of ~82%. This comment classification output from our models can be used to provide restaurants the summary of their performance, and it also helps to know how well they are doing. Future work can focus on more specific aspects of comment emotions (happy, angry, cool, etc.) and adding neutral as one of the labels (positive, negative, neutral).

*7.2 Deployment Procedures*

The best neural network model can be deployed to web server with the following steps:

* Saving current state of the best trained model
* Develop a web application using popular Flask web framework
* Deploy a machine learning application to a public web server