Sentiment Analysis to Indicate Customer Satisfaction
Executive Summary

D214 Performance Assessment (NKM2) Task 3

Corey B. Holstege

December 3, 2023

Table of Contents

Problem Statement and Hypothesis

What's in a word? Potentially much when it's a social media post that could go viral.

The internet was invented in 1983 (Board of Regents) and social media took off in 2003 with the launch of Myspace(Wikimedia Foundation, 2023, *Timeline of social media*). With those events the number of words exploded. According to Influence MarketingHub there are currently over 116 unique social media platforms with over 4.89 billion users – that is a lot of words.

What does this mean for today's consumer company? There are more ways than ever for customers to interact with the businesses they spend their money with. With all these different platforms how are companies to know if their customers are satisfied or upset with them overall? Afterall, to keep customers coming back and making further purchases they need to keep their customers happy. A key metric companies track is the customer satisfaction score (CSAT). This metric is calculated by taking a count of all of reviews where the company was rated a 4 or 5 (satisfied or very satisfied) divided by the total number of reviews, multiplied by 100 to turn it into a percent. The benchmark for fast food restaurants for CSAT is 76% (SurveyMonkey).

How do companies take all these reviews, in text format, and turn it into a CSAT? This is where neural networks for sentiment analysis shine. Like all machine learning algorithms, neural networks require a training data set – a set of data (reviews) that are already classified as "satisfied" or "dissatisfied" to train the model. Fortunately, companies already have this – anyplace customers leave reviews with one-to-five-star rating. One to three stars can be considered dissatisfied, with four and five stars as satisfied. These reviews could be left on their own website, on products they sell, or on third-party websites such as Consumer Reports. Once a model is completed that can accurately predict an input (review) as "satisfied" or "dissatisfied", the companies can then take all of their reviews from all of the social media platforms they are on and enter then into the model to classify the review. After that, it's simply calculating the CSAT score.

This is the focus of this project: can a neural network model be constructed on the dataset to accurately predict customer reviews as positive or negative, allowing these predictions to be used to calculate a customer satisfaction score?

Success of the project will be measured by accepting either the null hypothesis or the alternative hypothesis. The null hypothesis is that a neural network ***cannot*** be constructed from the dataset to accurately predict customer reviews as positive or negative. The alternative hypothesis is that a neural network ***can*** be constructed from the dataset to accurately predict customer reviews as positive or negative with an accuracy greater than eighty percent (80%).

Summary of the Data Analysis Process

The data for the model was extensively analyzed and then cleaned to prepare it for modeling. The information in the main Dataframe, df, that is required for modeling was split into a new dataframe df_reviews.

First, the "rating" column is cleaned by mapping the five unique values to 0 or 1:

```
# %%%%[7.2.1] PREPROCESS COLUMN: RATING
# create a dictionary to map to reivews ratings to 0 (negative sentiment) or 1 (positive sentiment)
rating_mapping = {'1 star': 0,
                  '2 stars': 0,
                  '3 stars': 0,
                  '4 stars': 1,
                  '5 stars': 1}
df_reviews['rating'] = df_reviews['rating'].map(rating_mapping)  # update the column in df_reviews
print(eda_analysis(df_reviews, 'rating'))
```

Next, the "review_time" column is cleaned by mapping the values to year/month/day values stored in a datetime format:

```
In [387]: time_mapping = {'review_time':
     ...:                 {'6 hours ago': '2023-10-01',
     ...:                     '8 hours ago': '2023-10-01',
     ...:                     '20 hours ago': '2023-10-01',
     ...:                     '21 hours ago': '2023-10-01',
     ...:                     '22 hours ago': '2023-10-01',
     ...:                     '23 hours ago': '2023-10-01',
     ...:                     'a day ago': '2023-10-01',
     ...:                     '2 days ago': '2023-10-01',
     ...:                     '3 days ago': '2023-10-01',
     ...:                     '4 days ago': '2023-10-01',
     ...:                     '5 days ago': '2023-10-01',
     ...:                     '6 days ago': '2023-10-01',
     ...:                     'a week ago': '2023-10-01',
     ...:                     '2 weeks ago': '2023-10-01',
     ...:                     '3 weeks ago': '2023-10-01',
     ...:                     '4 weeks ago': '2023-10-01',
     ...:                     'a month ago': '2023-09-01',
     ...:                     '2 months ago': '2023-08-01',
     ...:                     '3 months ago': '2023-07-01',
     ...:                     '4 months ago': '2023-06-01',
     ...:                     '5 months ago': '2023-05-01',
     ...:                     '6 months ago': '2023-04-01',
     ...:                     '7 months ago': '2023-03-01',
     ...:                     '8 months ago': '2023-02-01',
     ...:                     '9 months ago': '2023-01-01',
     ...:                     '10 months ago': '2022-12-01',
     ...:                     '11 months ago': '2022-11-01',
     ...:                     'a year ago': '2022-10-01',
     ...:                     '2 years ago': '2021-10-01',
     ...:                     '3 years ago': '2020-10-01',
     ...:                     '4 years ago': '2019-10-01',
     ...:                     '5 years ago': '2018-10-01',
     ...:                     '6 years ago': '2017-10-01',
     ...:                     '7 years ago': '2016-10-01',
     ...:                     '8 years ago': '2015-10-01',
     ...:                     '9 years ago': '2014-10-01',
     ...:                     '10 years ago': '2013-10-01',
     ...:                     '11 years ago': '2012-10-01',
     ...:                     '12 years ago': '2011-10-01'}}

In [388]: df_reviews.replace(time_mapping, inplace=True)  # replace the values in the column using the dictionary defined above

In [389]: df_reviews['review_time'] = pd.to_datetime(df_reviews['review_time'])  # change the datatype of the column
```

This column ended up not be used in the final analysis.

Finally, and most importantly, the "review" column is cleaned and prepared via a function called clean_review: punctation is removed; all text is converted to lowercase; stop words are removed; the character strings "xbf", "xef", and "xfd" are removed; reviews are tokenized, and reviews are padded to the same length.

Note this last step was done twice: padded once to full length (273) in column padded_sequences_max, and once to average length (6) in column padded_sequences_median.

```
In [387]: time_mapping = {'review_time':
     ...:                 {'6 hours ago': '2023-10-01',
     ...:                     '8 hours ago': '2023-10-01',
     ...:                     '20 hours ago': '2023-10-01',
     ...:                     '21 hours ago': '2023-10-01',
     ...:                     '22 hours ago': '2023-10-01',
     ...:                     '23 hours ago': '2023-10-01',
     ...:                     'a day ago': '2023-10-01',
     ...:                     '2 days ago': '2023-10-01',
     ...:                     '3 days ago': '2023-10-01',
     ...:                     '4 days ago': '2023-10-01',
     ...:                     '5 days ago': '2023-10-01',
     ...:                     '6 days ago': '2023-10-01',
     ...:                     'a week ago': '2023-10-01',
     ...:                     '2 weeks ago': '2023-10-01',
     ...:                     '3 weeks ago': '2023-10-01',
     ...:                     '4 weeks ago': '2023-10-01',
     ...:                     'a month ago': '2023-09-01',
     ...:                     '2 months ago': '2023-08-01',
     ...:                     '3 months ago': '2023-07-01',
     ...:                     '4 months ago': '2023-06-01',
     ...:                     '5 months ago': '2023-05-01',
     ...:                     '6 months ago': '2023-04-01',
     ...:                     '7 months ago': '2023-03-01',
     ...:                     '8 months ago': '2023-02-01',
     ...:                     '9 months ago': '2023-01-01',
     ...:                     '10 months ago': '2022-12-01',
     ...:                     '11 months ago': '2022-11-01',
     ...:                     'a year ago': '2022-10-01',
     ...:                     '2 years ago': '2021-10-01',
     ...:                     '3 years ago': '2020-10-01',
     ...:                     '4 years ago': '2019-10-01',
     ...:                     '5 years ago': '2018-10-01',
     ...:                     '6 years ago': '2017-10-01',
     ...:                     '7 years ago': '2016-10-01',
     ...:                     '8 years ago': '2015-10-01',
     ...:                     '9 years ago': '2014-10-01',
     ...:                     '10 years ago': '2013-10-01',
     ...:                     '11 years ago': '2012-10-01',
     ...:                     '12 years ago': '2011-10-01'}}

In [388]: df_reviews.replace(time_mapping, inplace=True)  # replace the values in the column using the dictionary defined above

In [389]: df_reviews['review_time'] = pd.to_datetime(df_reviews['review_time'])  # change the datatype of the column
```

Here is an example of how a review is cleaned and it's final output for modeling:

Column	Value
Review	Why does it look like someone spit on my food? I had a normal transaction, everyone was chill and polite, but now i dont want to eat this. Im trying not to think about what this milky white/clear substance is all over my food, i d*** sure am not coming back.
Review_lowercase	why does it look like someone spit on my food? i had a normal transaction, everyone was chill and polite, but now i dont want to eat this. im trying not to think about what this milky white/clear substance is all over my food, i d*** sure am not coming back.
Review_no_punct	why does it look like someone spit on my food i had a normal transaction everyone was chill and polite but now i dont want to eat this im trying not to think about what this milky white clear substance is all over my food i d sure am not coming back
Review_no_stopwords	look someone spit food normal transaction everyone chill polite now dont want eat im trying think milky white clear substance food d sure coming back
Review_no_xbf_xef	look someone spit food normal transaction everyone chill polite now dont want eat im trying think milky white clear substance food d sure coming back

Review_tokenized	[326, 294, 2196, 7, 595, 3021, 354, 2369, 505, 210, 337, 159, 104, 661, 444, 289, 3602, 938, 1219, 4782, 7, 572, 258, 345, 74]
Review_padded_max	[326, 294, 2196, 7, 595, 3021, 354, 2369, 505, 210, 337, 159, 104, 661, 444, 289, 3602, 938, 1219, 4782, 7, 572, 258, 345, 74, 0]
Review_padded_median	[4782, 7, 572, 258, 345, 74]

Of the four models evaluated, model 3 had the highest accuracy score at 87% and the lowest loss score at 33%. Model 3 has the following architecture:

- Keras sequential model
- Embedding parameters of:
  - Vocabulary size (total number of unique words in the dataset) of 15,232 words
  - Output dimension (4^{th} square root of the vocab size) of 11
  - Input length (number of tokens to consider) of 273 (which is how many words the longest review post cleaning is)
- Three dense, or hidden layers:
  - First dense layer has 100 nodes and uses the rectified linear unit, or RELU[1], activation function.
  - The second dense layer has 50 notes and uses the RELU activation function.
  - The third dense layer has 25 nodes and uses the RELU activation function.
- The final layer is a dense layer with 2 nodes, as the output for this model is either negative sentiment (0) or positive sentiment (1) with the softmax[2] activation function. The function returns two numbers, or probabilities, that sum to 1 and provide the probabilities that the text is either negative or positive sentiment.
- Finally, the model is complied using the "adam" optimizer[3], a loss parameter of "spare categorical crossentropy", and "accuracy" for the metrics.

```
In [433]: model_3 = Sequential()
     ...: model_3.add(Embedding(input_dim=15232, output_dim=11, input_length=273))
     ...: model_3.add(Flatten())  # https://keras.io/api/layers/reshaping_layers/flatten/
     ...: model_3.add(Dense(100, activation='relu'))
     ...: model_3.add(Dense(50, activation='relu'))
     ...: model_3.add(Dense(25, activation='relu'))
     ...: model_3.add(Dense(2, activation='softmax'))
     ...: model_3.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     ...: print(model_3.summary())
```

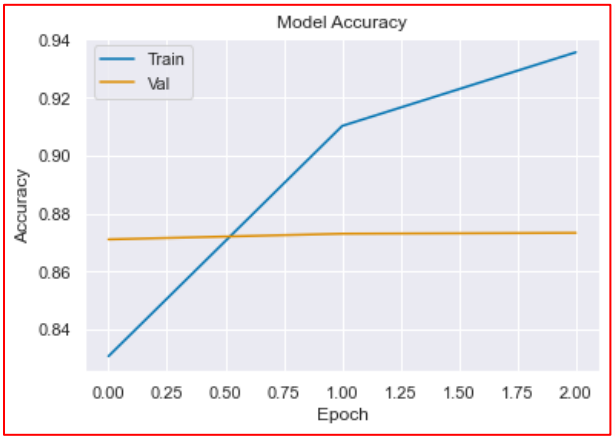
The model has a total of 474,329 parameters:

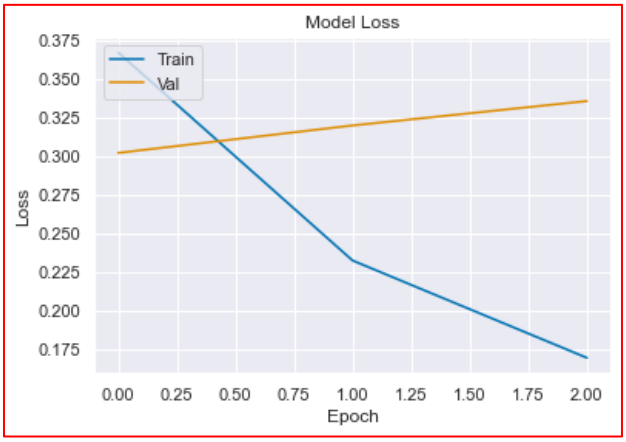
[1] https://keras.io/api/layers/activations/#relu-function
[2] https://keras.io/api/layers/activations/#softmax-function
[3] https://keras.io/api/optimizers/adam/

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_13 (Embedding)    (None, 273, 11)           167552

 flatten_13 (Flatten)        (None, 3003)              0

 dense_40 (Dense)            (None, 100)               300400

 dense_41 (Dense)            (None, 50)                5050

 dense_42 (Dense)            (None, 25)                1275

 dense_43 (Dense)            (None, 2)                 52

=================================================================
Total params: 474,329
Trainable params: 474,329
Non-trainable params: 0
_________________________________________________________________
None
```

Model 3 is fit on the training set with 93% accuracy:

```
In [434]: model_3_history = model_3.fit(X_train, y_train, epochs=20, batch_size=32, callbacks=[early_stopping_monitor], verbose=True,
validation_data=(X_test, y_test))
Epoch 1/20
832/832 [==============================] - 8s 9ms/step - loss: 0.3669 - accuracy: 0.8308 - val_loss: 0.3020 - val_accuracy: 0.8711
Epoch 2/20
832/832 [==============================] - 5s 6ms/step - loss: 0.2323 - accuracy: 0.9102 - val_loss: 0.3197 - val_accuracy: 0.8730
Epoch 3/20
832/832 [==============================] - 4s 5ms/step - loss: 0.1693 - accuracy: 0.9356 - val_loss: 0.3355 - val_accuracy: 0.8733

In [435]: print(model_3_history.history)
{'loss': [0.3669384717941284, 0.23229819536209106, 0.16931922733783722], 'accuracy': [0.8307784795761108, 0.9102044105529785, 0.9356026649475098],
'val_loss': [0.3019617199897766, 0.3197106122970581, 0.33551594614982605], 'val_accuracy': [0.871074378490448, 0.8730278015136719,
0.873283281326294]}
```





Then the model is evaluated on the training set with 95% accuracy:

```
In [437]: model_3_score = model_3.evaluate(X_train, y_train, verbose=0)

In [438]: print(f'Training Set: Test Loss: {model_3_score[0]} / Test Accuracy: {model_3_score[1]}')
Training Set: Test Loss: 0.12202907353639603 / Test Accuracy: 0.9557409286499023
```

Finally, the model is evaluated on the test set with 87% accuracy:

```
In [439]: model_3_evaluation = model_3.evaluate(X_test, y_test)
208/208 [==============================] - 1s 4ms/step - loss: 0.3355 - accuracy: 0.8733

In [440]: print(f'Test Set: Test Loss: {model_3_evaluation[0]} / Test Accuracy: {model_3_evaluation[1]}')
Test Set: Test Loss: 0.33551594614982605 / Test Accuracy: 0.8733283281326294
```

Finally, Thirty-three customer comments were manually scraped from McDonald's social media pages on Facebook, Instagram, LinkedIn, and Pinterest. These reviews are loaded into Dataframe df_sm, the clean_review function is applied to clean and prepare the data, and then model 3 is executed on these reviews:

```
In [471]: df_sm_predictions = model_3.predict(df_sm_padded_max)
2/2 [==============================] - 0s 5ms/step

In [472]: df_sm_predictions_analysis = pd.DataFrame(df_sm_predictions)  # convert to a dataframe
```

As these reviews do not have a rating score, they are spot checked to determine accuracy. As explained above, the softmax activation function returns a probability the review is negative (0) and a probability the review is positive (1) with both values summing to 1. For example, df_sm review at index 0 has a 98% chance of being negative, and a 1% chance of being positive. Review at index 1 has a 49% chance of being negative and 50% chance of being positive:

	0	1
0	0.980199	0.019801
1	0.49105	0.50895
2	0.868522	0.131478
3	0.723241	0.276759

For this analysis, we simply take the larger of the 2 numbers and use that to classify the review as positive or negative.

Outline of Findings

As shown above, model 3 has an accuracy of 87% with a loss rate of 33%:

```
In [439]: model_3_evaluation = model_3.evaluate(X_test, y_test)
208/208 [==============================] - 1s 4ms/step - loss: 0.3355 - accuracy: 0.8733

In [440]: print(f'Test Set: Test Loss: {model_3_evaluation[0]} / Test Accuracy: {model_3_evaluation[1]}')
Test Set: Test Loss: 0.33551594614982605 / Test Accuracy: 0.8733283281326294
```

Predications on df_sm reviews were spot checked and as a whole appear to be accurate. Appears accurate as a negative review:

```
In [475]: num_index = 23

In [476]: print('Origional review', df_sm['review'][num_index], '\n')
     ...: print('Predicted:', 'Negative' if df_sm_predictions[num_index][0] >= 0.5 else 'Positive', 'review')
Origional review Yesterday around 18.00, I bought a Double Big Mc for my son from your place in Yalova (Turkey) center. My son is autistic and
doesn't eat anything other than meatballs and cheese. I said nothing should be added except cheese. They said okay. I warned them once more before
the package arrived, and they called inside again. I bought the package and brought it home and there was something resembling grated onion on the
meatball and my son did not eat it. I became very angry. This isn't the first time this has happened to me. On average, they make a mistake every
3-4 times in your same store. Moreover, there was no intensity. I'm too afraid to buy anything from you now. I will express your insensitivity on
every platform.

Predicted: Negative review
```

Appears to be classified incorrectly:

```
In [477]: num_index = 16

In [478]: print('Origional review', df_sm['review'][num_index], '\n')
     ...: print('Predicted:', 'Negative' if df_sm_predictions[num_index][0] >= 0.5 else 'Positive', 'review')
Origional review dear grimace and mcdonalds, why must i purchase a whole meal in order to receive a grimace shake from your establishments?
sometimes i am not so hungry that i would eat a whole big mac and fry/ 10 pc chicken mcnugget and fry, but I still might crave the bombastic and
powerful flavor of a grimace\x92s birthday shake. why would you engage in such non consumer friendly practices and marketing tactics????

Predicted: Positive review
```

Appears to be classified correctly as positive:

```
In [482]: print('Origional review', df_sm['review'][num_index], '\n')
     ...: print('Predicted:', 'Negative' if df_sm_predictions[num_index][0] >= 0.5 else 'Positive', 'review')
Origional review McDonald\x92s is one of my favorite burgers

Predicted: Positive review
```

Finally, we prove out that customer satisfaction can be calculated:
The total number of reviews from the Kaggle dataset that are positive and the total number of reviews from the manually scaped social media posts that are positive are calculated and stored in the variables csat_satisfied_df_reviews and csat_satisfied_df_sm respectively.

The total number of reviews from the Kaggle dataset are calculated and the total number of reviews from the manually scraped social media posts are calculated and stored in the variables csat_all_df_reviews and csat_add_df_sm respectively.

A percentage is calculated by summing the two counts of satisfied reviews, dividing by the sum of the two total counts, and multiplying by 100.

```
In [483]: csat_satisfied_df_reviews = sum(df_reviews['rating'] == 1)

In [484]: csat_satisfied_df_sm = sum(df_sm['model_3_prediction'] == 1)

In [485]: csat_all_df_reviews = len(df_reviews['rating'])

In [486]: csat_all_df_sm = len(df_sm['model_3_prediction'])

In [487]: csat = (csat_satisfied_df_reviews + csat_satisfied_df_sm) / (csat_all_df_reviews + csat_all_df_sm) * 100

In [488]: print(f'CSAT score: {round(csat,2)}%')
CSAT score: 48.01%
```

The customer satisfaction score for McDonald's for this dataset is 48%.

Explanation of the Limitations of Techniques and Tools Used

The biggest limitation of this analysis is the size of the dataset. At only 33,396 reviews for the dataset from Kaggle, and only 33 reviews for social media posts the data size is tiny. This data is enough to prove out the concept; however, further tuning on the model would be required with larger, real-world datasets. This was the largest dataset that could be reasonably obtained

for this project. Companies leveraging business-to-business API's could obtain greater data with ease.

Summary of Proposed Actions

This project serves as a proof of concept and proved the following:

- Company datasets of reviews where customers provide a star or 1-to-5 rating are customers classifying their sentiment of the review.
- These datasets can be used to train a neural network and provide a model.
- This model can then be used to classify the sentiment of other text from social media platforms as positive (1) or negative (0).
- Collectively all these ratings can be used to provide a Customer Satisfaction score.

A few next steps that should be taken:

1) The proof of concept should be further proved by gathering more reviews with star ratings to use to train the model, and more social media reviews should be gathered to be evaluated. At 33,396 reviews the dataset is small for a neural network.
2) The model should be hyper tuned to determine the optimal input parameters. For example, models were executed using an input length of 273 tokens (the max review length) or 6 tokens (the average review length) with close accuracy results between these two. However, different input lengths between 6 and 273 were not tested. Reducing the input length for 273 down to some lower number will reduce the vector database size required for this to be executed at scale.
3) The closer to real time this analysis can get, the more useful it will be. After the above steps are completed and an optimal model obtained, a pipeline should be set up to ingest reviews from social media sites via their API's, evaluate the sentiment of the reviews with the model, and calculate customer satisfaction. This data should be assigned to store locations and provided in a dashboard format so store leadership can see customer satisfaction scores in near real time, with review text. This will allow store leadership to make immediate actionable decisions to improve or maintain customer satisfaction.

Expected Benefits of the Study

With the speed of the internet and influencers, a written word can have an outsized impact faster than ever before. This project provides a means and method to collect all those written words, analyze them as positive or negative, and feed that information into a commonly understood metric: customer satisfaction score.

Taken in aggregate, this metric informs company leadership at a wholistic level how the company is perceived. Paired with location data it brings this insight down to the store level where local store leadership can immediately review and act upon it.

Once implemented in a production landscape and paired with a dashboard this model is expected to provide the near real time feedback required to increase customer satisfaction which will ultimately drive repeat and incremental sales.

Appendix

Sources:

Board of Regents of the University System of Georgia. (n.d.). *A Brief History of the Internet*. Online Library Learning Center. Retrieved October 28, 2023, from: https://www.usg.edu/galileo/skills/unit07/internet07_02.phtml#:~:text=January%201%2C%201983%20is%20considered,Protocol%20(TCP%2FIP)a

SurveyMonkey. *The Ultimate Guide to Customer Satisfaction Score*. SurveyMonkey. (n.d.). Retrieved October 28, 2023, from: https://www.surveymonkey.com/resources/premium/customer-satisfaction-score-csat-guide/

Wikimedia Foundation. (2023a, October 14). *Timeline of social media*. Wikipedia. Retrieved October 28, 2023, from: https://en.wikipedia.org/wiki/Timeline_of_social_media

Datacamp: Introduction to Deep Learning in Python
D213 Task 2 Cohort Webinar PPT.pptx

End of document.