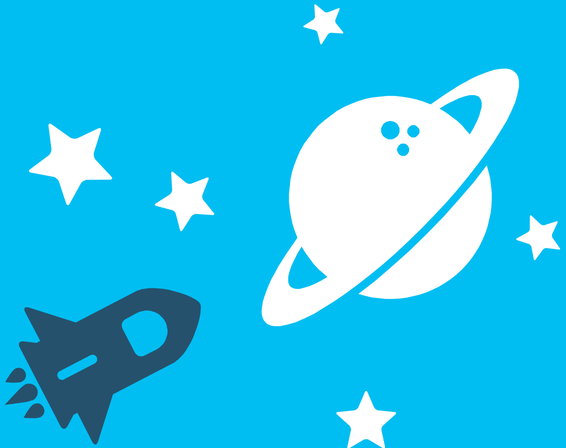
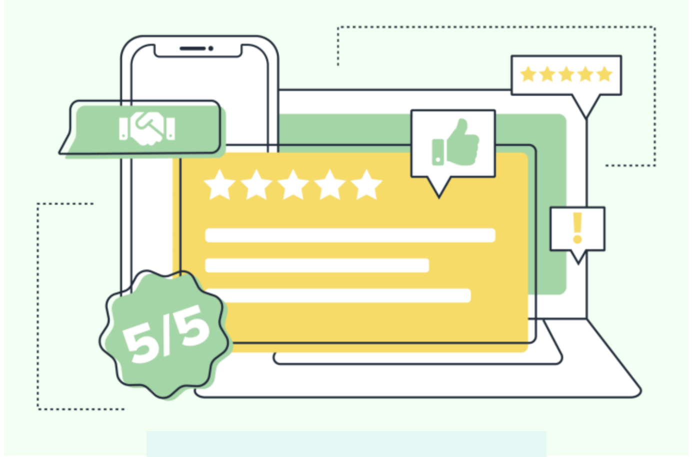
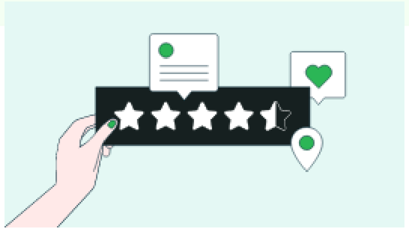
Sentiment Analysis of a restaurant using Deep RNN Techniques



PROBLEM DEFINITION

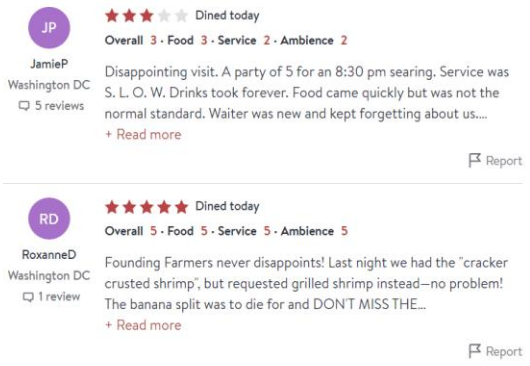
- Analyze a prominent restaurant's review from Opentable.
- Electronic word-of-mouth (WoM) platforms
- More reliable and trustworthy than advertisements from retailers.
- Traditionally, people take suggestions from acquaintances, friends, and family.
- Customer can also share their feedback and view other people’s feedback about restaurant service, food, taste, ambiance, and price
- Helps put the restaurant on the map.





PROBLEM DEFINITION

- Founding Farmers (3 branches).
- Scrape all the reviews from the website and label the sentiments based on their polarity.
- We fit the data into three different RNN models and compare their performance.
- Finally, we test the best model's performance with an User Generated Input.



JP
JamieP
Washington DC
5 reviews

★★★☆☆ Dined today
Overall 3 · Food 3 · Service 2 · Ambience 2

Disappointing visit. A party of 5 for an 8:30 pm searing. Service was S. L. O. W. Drinks took forever. Food came quickly but was not the normal standard. Waiter was new and kept forgetting about us....
+ Read more

Report

RD
RoxanneD
Washington DC
1 review

★★★★★ Dined today
Overall 5 · Food 5 · Service 5 · Ambience 5

Founding Farmers never disappoints! Last night we had the "cracker crusted shrimp", but requested grilled shrimp instead—no problem! The banana split was to die for and DON'T MISS THE...
+ Read more

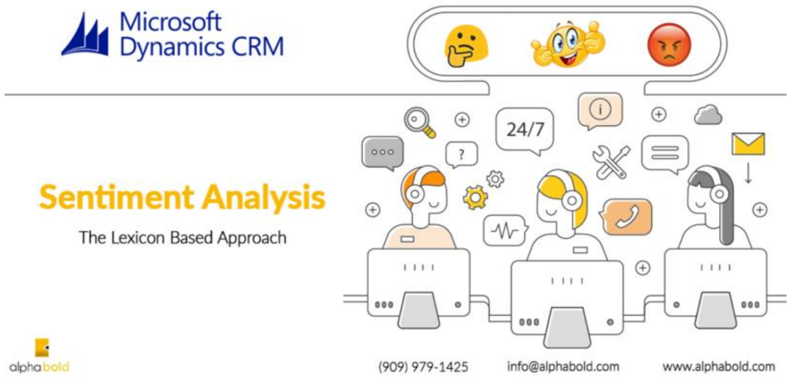
Report

RELATED WORKS!

- Similar sentimental Analysis for reviews of Amazon, TripAdvisor and IMDB.
- LSTM model used show better sentiment classification of 85% accuracy, using CNN before LSTM helps to identify important features, reduces training time.
- No need for complex feature engineering, avoids domain specific expertise.

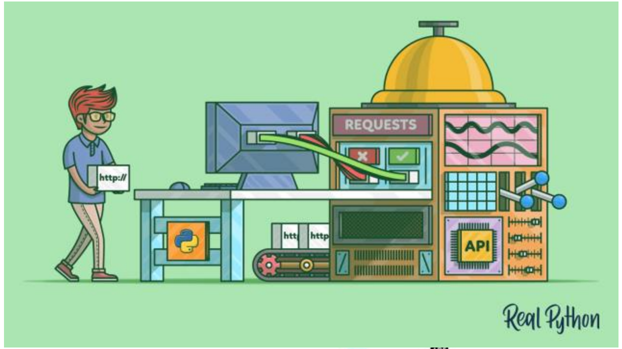
RELATED WORKS!

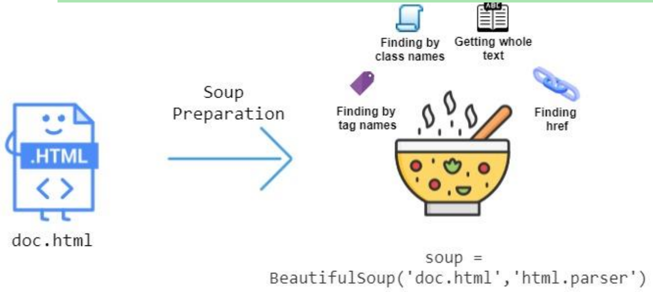
- Sentimental Analysis of tweets was done by using pre trained word embedding model.
- Research team focused on sentiment lexicon W-WSD, SentiWordNet, Textblob.
- Dynamics 365 Customer Insights offers a sentiment analysis model for customer feedback that enables organizations to analyze their data. It assigns each feedback comment a sentiment score and associates each feedback with all relevant business aspects.



DATA PREPARATION: Part-1 (Getting the Data)

- Python Requests module to request the website where the reviews are present
- BeautifulSoup to traverse the result to extract the required parts.
- Convert the request result to a BeautifulSoup object (aka making the Soup).
- Convert raw uncleaned reviews into a data frame for pre-processing and analyzing the data.

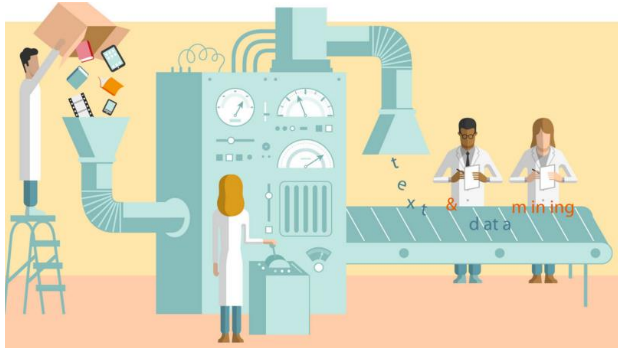


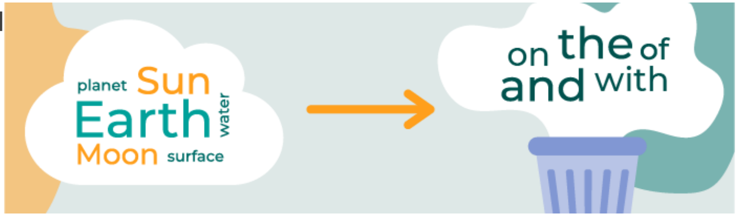


```
soup =
BeautifulSoup('doc.html','html.parser')
```

DATA PREPARATION: Part - 2 (Data Cleaning)

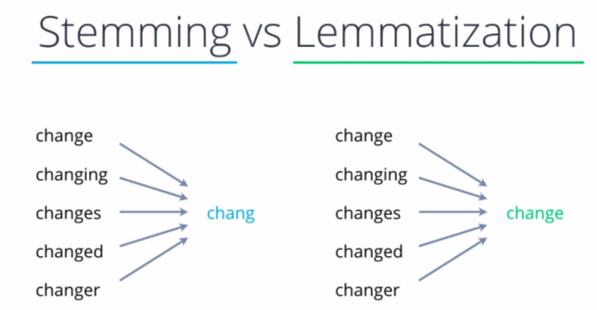
- The four most common approaches lowercasing all words, removing punctuation, removing Stopwords, and removing trivial words.
- Performed word tokenization using Keras, we use the text_to_word_sequence method.
- Used the NLTK module to calculate the number of Stopword counts per review; Stopwords are commonly occurring words and do not contribu the sentiment analysis.
- Remove the outliers.





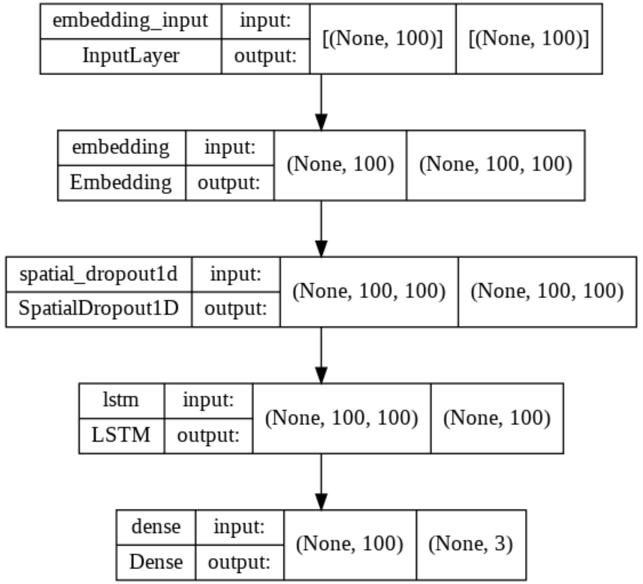
DATA PREPARATION: Part - 3 (Sentiment Analysis)

- Lemmatization: Transforming your natural text from English to lemming language.
- Cuts out the number of words that are available for analysis by combining similar forms into one base form
- 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
- Cleaned and Labelled for Model fitting.



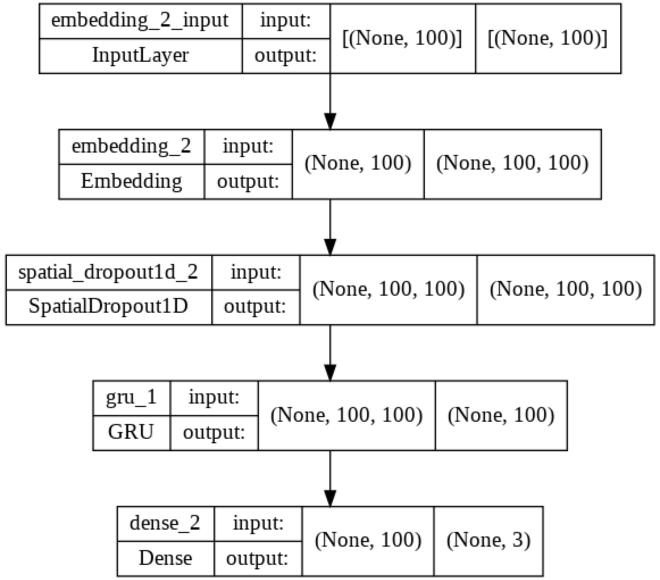
METHODS APPLIED - LSTM

- Convert the positive integer representation of words to word embeddings through a embedding layer
- Map each word to a real valued vector of 100, limit total words of interest to 50,000 common words, limit each review to 100
- Add sequential model, a SpatialDropout 1D of 0.4 is added
- Add LSTM layer with 100 units dimension of output space, hyperparameters - dropout of 0.4
- Dense layer of 3, “softmax” activation function.


embedding_input InputLayer | input: [(None, 100)] | output: [(None, 100)]
embedding Embedding | input: (None, 100) | output: (None, 100, 100)
spatial_dropout1d SpatialDropout1D | input: (None, 100, 100) | output: (None, 100, 100)
lstm LSTM | input: (None, 100, 100) | output: (None, 100)
dense Dense | input: (None, 100) | output: (None, 3)

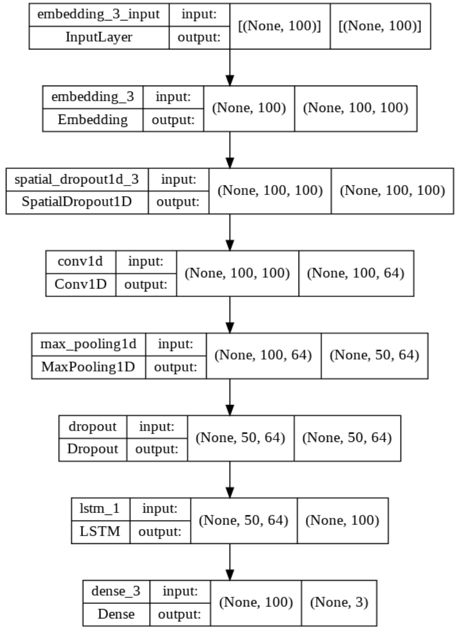
METHODS APPLIED - GRU

- Add an embedding layer, real values vector of 100.
- Add a SpatialDropout 1D of 0.4
- Add a GRU Layer, giving 100 units to the dimensions of output space, hyperparameters being dropout - 0.4
- Add a dense layer of 3, activation function is "softmax"



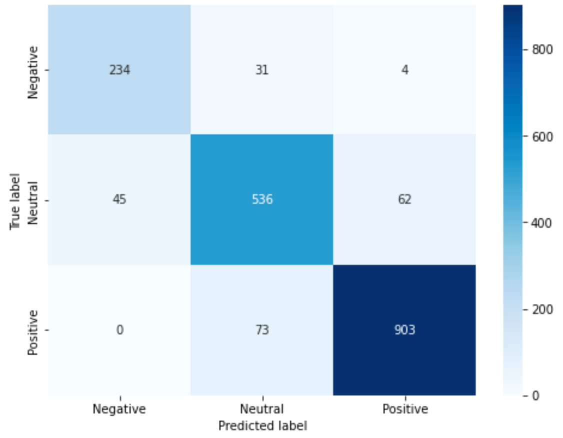
METHODS APPLIED - (CNN-LSTM Hybrid)

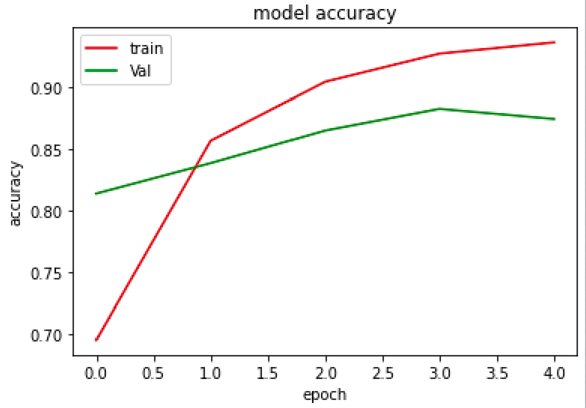
- Add a embedding layer, hyper parameters are the largest data features - 100 embedding dimensions.
- SpatialDropout 1D of 0.4
- Add a convolution layer, with hyperparameters being 64 filters, dimensionality of output space. Kernel size of 5 and “same” padding. Relu activation function.
- Add a Max Pooling layer 1D of size 2, dropout of 0.4.
- The output of this layer is then added as an input to the LSTM layer with 100 units and dropout of 0.4
- Results in a dense output layer of 3 outputs, activation function “softmax”


embedding_3_input InputLayer input: output: [(None, 100)] [(None, 100)]
embedding_3 Embedding input: output: (None, 100) (None, 100, 100)
spatial_dropout1d_3 SpatialDropout1D input: output: (None, 100, 100) (None, 100, 100)
conv1d Conv1D input: output: (None, 100, 100) (None, 100, 64)
max_pooling1d MaxPooling1D input: output: (None, 100, 64) (None, 50, 64)
dropout Dropout input: output: (None, 50, 64) (None, 50, 64)
lstm_1 LSTM input: output: (None, 50, 64) (None, 100)
dense_3 Dense input: output: (None, 100) (None, 3)

EXPERIMENTS - LSTM

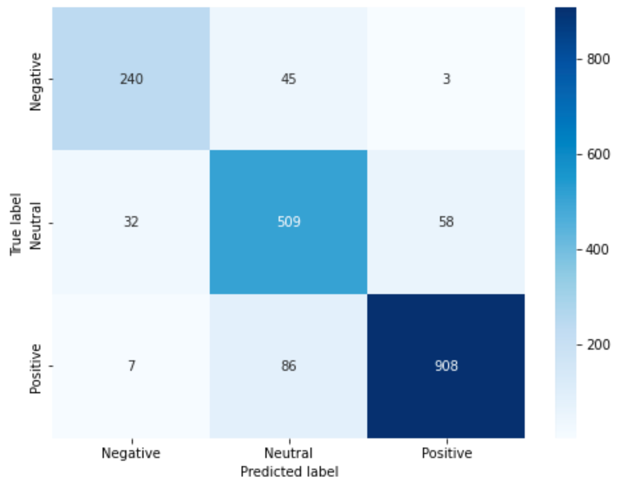
- Obtained an accuracy score for testing is 88%
- Accuracy score of 93% on training set
- Epochs 5, F1 score was 88.6%



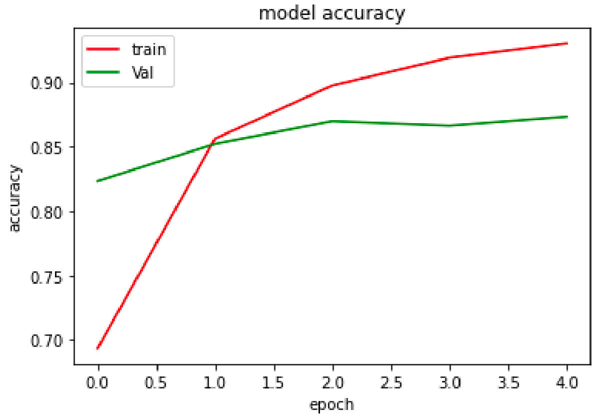


EXPERIMENTS - GRU

- Accuracy on testing set is 87%
- Training set accuracy is 93%
- Epochs 5, F1 score of 87.85%

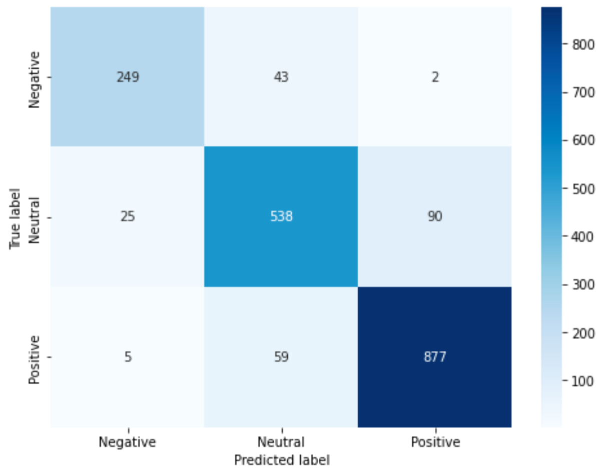


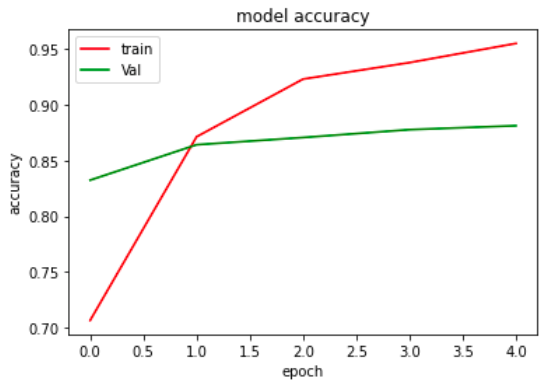
True label \ Predicted label	Negative	Neutral	Positive
Negative	240	45	3
Neutral	32	509	58
Positive	7	86	908



EXPERIMENTS - (CNN-LSTM Hybrid)

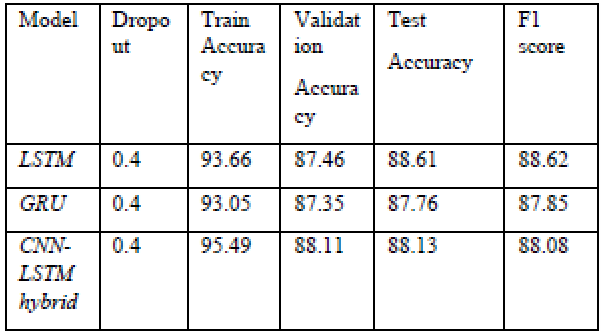
- Testing set accuracy was 88.13%
- Training set accuracy was 95.49%
- Epochs 5, F1 score of 88.08%





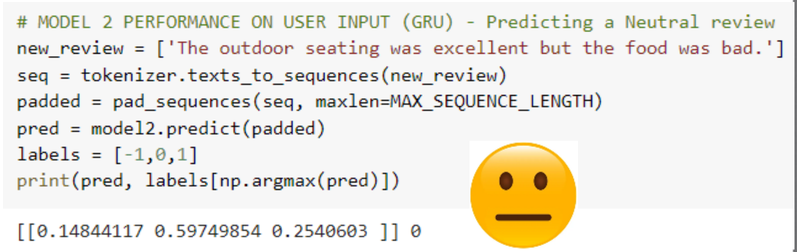
CONCLUSION

- We could see that the models' performance doesn't vary much between themselves regarding the model accuracy and prediction. All the models performed well.
- We selected both the GRU and LSTM models for testing them with user-generated reviews since they have relatively low gap between Training and Validation Accuracy.
- The most optimum model from our project would be the GRU model since it has just one layer and uses less training parameter and therefore uses less memory and executes faster than LSTM
- Future ideas that could be added to the current model can be an auto correction function in the preprocessing stage as not all reviews would be correctly spelled.
- Scrape more data



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



```
# MODEL 2 PERFORMANCE ON USER INPUT (GRU) - Predicting a Neutral review
new_review = ['The outdoor seating was excellent but the food was bad.']
seq = tokenizer.texts_to_sequences(new_review)
padded = pad_sequences(seq, maxlen=MAX_SEQUENCE_LENGTH)
pred = model2.predict(padded)
labels = [-1,0,1]
print(pred, labels[np.argmax(pred)])

[[0.14844117 0.59749854 0.2540603 ]] 0
```