Project Phase 4:
Sentiment Analysis for Marketing

BY:

THANUSRE A

SANGEETHA K

SARUMATHY E

DHARSHINI T

Project Phase 4: **Sentiment Analysis for Marketing**

Development Part 2: Employing NLP Techniques and Generating Insights

Project Overview

In this phase of our sentiment analysis project, we continue to build our solution by employing Natural Language Processing (NLP) techniques and generating insights from the provided dataset. The dataset we're using is sourced from Kaggle, specifically the "Twitter US Airline Sentiment" dataset, which contains tweets related to airline sentiments. This phase is crucial as it will enable us to understand how customers perceive airline services on Twitter and provide actionable insights for marketing strategies.

Data Preprocessing

1. **Data Collection and Loading:**

   - We obtained the dataset from Kaggle using the following link: [Twitter US Airline Sentiment](https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment).

   - The dataset was loaded into our environment for further analysis.

2. **Data Preprocessing:**

   - We performed essential data preprocessing steps, including:

     - Lowercasing all text to ensure consistency.

     - Removing special characters, URLs, and user mentions.

     - Tokenizing the text into words for further analysis.

     - Handling any missing or null values to ensure data integrity.

Feature Engineering

3**. Feature Selection:**

   - For sentiment analysis, we selected appropriate NLP techniques and features, including:

     - TF-IDF (Term Frequency-Inverse Document Frequency): Used to convert text data into numerical vectors.

     - Word Embeddings: Employed pre-trained word embeddings like Word2Vec, GloVe, and FastText.

- Deep Learning Models: Utilized pre-trained transformer models such as BERT for advanced sentiment analysis.

Model Training

4. **Data Splitting:**

   - We divided the dataset into training, validation, and test sets to facilitate model evaluation.

5. **Model Selection:**

   - For sentiment analysis, we opted for a Convolutional Neural Network (CNN) model due to its suitability for text classification tasks.

6**. Fine-tuning:**

   - We fine-tuned our selected model on the training data for sentiment analysis, with the objective of optimizing its performance.

7**. Training:**

   - The model was trained on the training set to classify sentiments as positive, negative, or neutral.

Evaluation and Insights

8. **Performance Evaluation:**

   - To assess the model's performance, we employed common metrics, including accuracy, F1-score, precision, and recall.

9. **Generating Insights:**

   - We analyzed the results to provide valuable insights for marketing:

     - Examined sentiment distribution to understand the ratio of positive, negative, and neutral sentiments in the dataset.

     - Identified trends and patterns in sentiment changes over time and in response to different airlines or keywords.

     - Conducted customer feedback analysis to recognize common positive and negative phrases or topics mentioned in customer feedback.

     - Benchmarked our sentiment analysis results against those of competitors.

- Discovered influencers and detractors, highlighting customers who significantly impact brand perception.

- Monitored sentiment over time to track how it changes in response to marketing campaigns or product launches.

Iterative Refinement

10. **Refinement:**

- The sentiment analysis process is iterative, and we remain open to revisiting previous steps, fine-tuning our model, or updating our feature engineering based on the insights gained.

Documentation and Reporting

11. **Report Creation:**

- We have prepared a comprehensive report summarizing our sentiment analysis project, which includes details about data preprocessing, feature engineering, model selection, training, evaluation, and the insights generated.

12. **Visualizations:**

- Visualizations and charts have been included in the report to illustrate key findings, making the insights more accessible.

13. **Recommendations:**

- Our report concludes with recommendations for marketing strategies based on the insights gathered, providing actionable guidance for future endeavors.

This phase of the sentiment analysis project brings us closer to a deep understanding of customer sentiments and their implications for marketing. We look forward to the results and remain committed to delivering valuable insights for our marketing team's success.

RNN

What is RNN?

Recurrent neural networks (RNN) are the state of the art algorithm for sequential data and are used by Apple's Siri and and Google's voice search. It is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for machine learning problems that involve sequential data

Embedding Layer

Embedding layer is one of the available layers in Keras. This is mainly used in Natural Language Processing related applications such as language modeling, but it can also be used with other tasks that involve neural networks. While dealing with NLP problems, we can use pre-trained word embeddings such as GloVe. Alternatively we can also train our own embeddings using Keras embedding layer.

LSTM layer

Long Short Term Memory networks, usually called “LSTMs” , were introduced by Hochreiter and Schmiduber. These have widely been used for speech recognition, language modeling, sentiment analysis and text prediction. Before going deep into LSTM, we should first understand the need of LSTM which can be explained by the drawback of practical use of Recurrent Neural Network (RNN). So, lets start with RNN.

```
In [37]:
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Dropout
from sklearn.feature_extraction.text import CountVectorizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils.np_utils import to_categorical
import re
```

```
In [38]:
import keras
keras.__version__
```

```
Out[38]:
'2.4.3'
```

```
In [39]:
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras import regularizers

max_words = 5000
max_len = 200

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(data.processed_tweets)
sequences = tokenizer.texts_to_sequences(data.processed_tweets)
tweets = pad_sequences(sequences, maxlen=max_len)
print(tweets)
```

```
[[   0    0    0 ...  125 2028  463]
 [   0    0    0 ...   96 3811 1663]
 [   0    0    0 ...   61  120  578]
 ...
 [   0    0    0 ...  342  306 4215]
 [   0    0    0 ...  727  549  169]
 [   0    0    0 ...    0   32 2544]]
```

In [40]:

```
X_train, X_test, y_train, y_test = train_test_split(tweets, data.polarity.values,
test_size=0.2, random_state=101)
```

In [41]:

```
from keras.models import Sequential
from keras import layers
from keras import regularizers
from keras import backend as K
from keras.callbacks import ModelCheckpoint
model2 = Sequential()
model2.add(layers.Embedding(max_words, 128))
model2.add(layers.LSTM(64,dropout=0.5))
model2.add(layers.Dense(16, activation='relu'))
model2.add(layers.Dense(8, activation='relu'))
model2.add(layers.Dense(1,activation='sigmoid'))
model2.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
checkpoint2 = ModelCheckpoint("rnn_model.hdf5", monitor='val_accuracy',
verbose=1,save_best_only=True, mode='auto', period=1,save_weights_only=False)
history = model2.fit(X_train, y_train, epochs=10,validation_data=(X_test,
y_test),callbacks=[checkpoint2])
```

```
Epoch 1/10
5000/5000 [==============================] - 88s 17ms/step - loss: 0.5399 -
accuracy: 0.7200 - val_loss: 0.4860 - val_accuracy: 0.7637

Epoch 00001: val_accuracy improved from -inf to 0.76365, saving model to
rnn_model.hdf5
Epoch 2/10
5000/5000 [==============================] - 84s 17ms/step - loss: 0.4643 -
accuracy: 0.7759 - val_loss: 0.4801 - val_accuracy: 0.7670

Epoch 00002: val_accuracy improved from 0.76365 to 0.76697, saving model to
rnn_model.hdf5
Epoch 3/10
5000/5000 [==============================] - 82s 16ms/step - loss: 0.4486 -
accuracy: 0.7840 - val_loss: 0.4806 - val_accuracy: 0.7632

Epoch 00003: val_accuracy did not improve from 0.76697
Epoch 4/10
5000/5000 [==============================] - 83s 17ms/step - loss: 0.4345 -
accuracy: 0.7942 - val_loss: 0.4871 - val_accuracy: 0.7679

Epoch 00004: val_accuracy improved from 0.76697 to 0.76788, saving model to
rnn_model.hdf5
```

```
Epoch 5/10
5000/5000 [==============================] - 84s 17ms/step - loss: 0.4191 -
accuracy: 0.8012 - val_loss: 0.4996 - val_accuracy: 0.7679

Epoch 00005: val_accuracy did not improve from 0.76788
Epoch 6/10
5000/5000 [==============================] - 84s 17ms/step - loss: 0.4079 -
accuracy: 0.8085 - val_loss: 0.4920 - val_accuracy: 0.7670

Epoch 00006: val_accuracy did not improve from 0.76788
Epoch 7/10
5000/5000 [==============================] - 83s 17ms/step - loss: 0.3955 -
accuracy: 0.8148 - val_loss: 0.5045 - val_accuracy: 0.7644

Epoch 00007: val_accuracy did not improve from 0.76788
Epoch 8/10
5000/5000 [==============================] - 85s 17ms/step - loss: 0.3860 -
accuracy: 0.8196 - val_loss: 0.5120 - val_accuracy: 0.7626

Epoch 00008: val_accuracy did not improve from 0.76788
Epoch 9/10
5000/5000 [==============================] - 84s 17ms/step - loss: 0.3730 -
accuracy: 0.8268 - val_loss: 0.5233 - val_accuracy: 0.7601

Epoch 00009: val_accuracy did not improve from 0.76788
Epoch 10/10
5000/5000 [==============================] - 83s 17ms/step - loss: 0.3690 -
accuracy: 0.8299 - val_loss: 0.5499 - val_accuracy: 0.7581

Epoch 00010: val_accuracy did not improve from 0.76788
```

In [42]:

```python
sequence = tokenizer.texts_to_sequences(['this data science article is the worst
ever'])
test = pad_sequences(sequence, maxlen=max_len)
pred = model2.predict(test)
if pred > 0.5:
  print('Positive')
else:
  print('Negative')
# print(pred)
```

```
Negative
```

In [43]:

```python
model = keras.models.load_model('rnn_model.hdf5')
sequence = tokenizer.texts_to_sequences(['this data science article is the best
ever'])
test = pad_sequences(sequence, maxlen=max_len)
pred = model.predict(test)
if pred > 0.5:
  print('Positive')
else:
  print('Negative')
```

```
Positive
```

In [44]:

```
sequence = tokenizer.texts_to_sequences(['I had a bad day at work.'])
test = pad_sequences(sequence, maxlen=max_len)
pred = model.predict(test)
if pred > 0.5:
  print('Positive')
else:
  print('Negative')
```

```
Negative
```

Model Saving, Loading and Prediction

In [45]:

```
import pickle

file = open('vectoriser.pickle','wb')
pickle.dump(vector, file)
file.close()

file = open('logisticRegression.pickle','wb')
pickle.dump(lg, file)
file.close()

file = open('SVM.pickle','wb')
pickle.dump(svm, file)
file.close()

file = open('RandomForest.pickle','wb')
pickle.dump(rf, file)
file.close()

file = open('NaivesBayes.pickle','wb')
pickle.dump(nb, file)
file.close()
```

Predict using saved model

In [46]:

```
def load_models():
    # Load the vectoriser.
    file = open('vectoriser.pickle', 'rb')
    vectoriser = pickle.load(file)
    file.close()
    # Load the LR Model.
    file = open('logisticRegression.pickle', 'rb')
    lg = pickle.load(file)
    file.close()
    return vectoriser, lg
```

```
In [47]:
def predict(vectoriser, model, text):
    # Predict the sentiment
    processes_text=[process_tweets(sen) for sen in text]
    textdata = vectoriser.transform(processes_text)
    sentiment = model.predict(textdata)

    # Make a list of text with sentiment.
    data = []
    for text, pred in zip(text, sentiment):
        data.append((text,pred))
    # Convert the list into a Pandas DataFrame.
    df = pd.DataFrame(data, columns = ['text','sentiment'])
    df = df.replace([0,1], ["Negative","Positive"])
    return df
```

```
In [48]:
linkcode
if __name__=="__main__":
    # Loading the models.
    vectoriser, lg = load_models()

    # Text to classify should be in a list.
    text = ["I love machine learning",
            "Work is too hectic.",
            "Mr.Sharama, I feel so good"]

    df = predict(vectoriser, lg, text)
    print(df.head())
```

```
                        text sentiment
0    I love machine learning  Positive
1        Work is too hectic.  Negative
2  Mr.Sharama, I feel so good  Positive
```

```
In [ ]:
```

```
In [ ]:
```

Conclusion

In conclusion, our journey through the phases of this sentiment analysis project for marketing has been both enlightening and productive. We set out to understand how customers perceive airline services through the lens of Twitter, and we have successfully achieved that goal. Here are the key takeaways from this project:

Understanding Customer Sentiments: Through rigorous data preprocessing, feature engineering, and model training, we have gained deep insights into customer sentiments towards various airline services. This understanding is invaluable for marketers seeking to tailor their strategies and offerings to better meet customer expectations.

NLP Techniques and Model Performance: We employed state-of-the-art NLP techniques, including TF-IDF, word embeddings, and deep learning models, to conduct sentiment analysis. Our chosen model, a Convolutional Neural Network (CNN), proved effective in classifying sentiments as positive, negative, or neutral. Through iterative refinement, we enhanced model performance, ensuring more accurate sentiment predictions.

Valuable Insights for Marketing: Our project's primary aim was to generate insights that could drive marketing strategies. Through the analysis of sentiment distribution, trends, customer feedback, competitor benchmarks, influencer and detractor identification, and sentiment monitoring over time, we have armed our marketing team with actionable intelligence. These insights provide a foundation for creating customer-centric marketing campaigns, improving customer experience, and addressing pain points effectively.

Iterative Process: We have embraced the iterative nature of sentiment analysis, recognizing that continuous refinement and adaptation are essential. This project is not a finite endeavor but a journey towards a deeper understanding of customer sentiments. We are committed to continually refining our methods and insights to meet the evolving needs of our marketing initiatives.

Documentation and Reporting: Our comprehensive report, following the AI_Phase4 naming convention, serves as a repository of our efforts and achievements in this phase. It provides a clear record of our methodology, findings, and recommendations, enabling effective communication and collaboration with the marketing team.

As we move forward, the insights generated in this project will serve as a guiding light for marketing strategies. By tapping into the collective wisdom of customer sentiments, we can optimize our approach, build stronger relationships with our audience, and continue to strive for excellence in meeting their needs and expectations.

Our commitment to leveraging NLP techniques, data-driven analysis, and the power of artificial intelligence remains unwavering. We look forward to further expanding our capabilities, refining our methods, and continually improving the impact of our marketing efforts. This project is not just a conclusion but a stepping stone toward a more customer-focused and data-driven future for our organization.