Sentiment Analysis of US Airlines Customer Feedback

Team Members
Aravind Reddy Apuri
Pranay Kumar Ganji
Sandeep Sriramula

Statement of Project Objectives

By analyzing customer feedback, such as survey responses and social media discussions, brands can pay heed to their customers and modify their products and services according to their preferences. Nevertheless, the opinionated information gathered from Twitter is in an unstructured textual format.

Unstructured data can be difficult, time-consuming, and costly to analyze, comprehend, and organize. Nevertheless, businesses can leverage sentiment analysis to automatically interpret, process, and label this data, enabling them to extract meaningful insights from unstructured text.

This project's goal is to conduct sentiment analysis on six US Airlines' tweets.

The deleted tweets from each customer's account express either positive, negative, or neutral opinions about the airline.

The task is to analyze how the travelers in February 2022 expressed their feelings on Twitter about six major US airlines

Statement of Value

The statement of this project is to build and compare different models for sentiment analysis on airline tweets. Specifically, we have implemented a LSTM model, a 1D CNN model, and a pre-trained BERT model. We have preprocessed the text data, tokenized the text, and trained the models on the tokenized data. We have evaluated the models based on their accuracy, precision, recall, and F1 score. The aim of the project is to identify the best model for sentiment analysis on airline tweets.

Review of State of Art

- The dataset used in the project is obtained from Kaggle and is focused on analyzing the sentiment of Twitter users towards six US airlines.
- The primary objective of the project is to use various algorithms to process this unstructured data and perform sentiment analysis on the tweets.
- Additionally, customer feedback is also being analyzed, and unstructured data is being processed to analyze the responses.

Overview of Dataset

- Dataset Link: Twitter US Airline Sentiment | Kaggle
- Features in dataset :
  - Number of Instances : 14640
  - Number of Classes : 3 (Positive, Negative, Neutral)
  - Number of Attributes : 15

Approach

1. **Data collection**: We collected airline tweets dataset from Twitter for sentiment analysis (From Kaggle).
2. **Data preprocessing**: We performed some preprocessing steps on the collected data, including removing URLs, converting to lowercase, removing special characters and punctuation marks, and removing stopwords.
3. T**okenization**: We tokenized the preprocessed text data using the Keras Tokenizer class, which converts the text into sequences of integers.
4. **Padding:** We added padding to the sequences to make them of equal length using the Keras pad_sequences() function.
5. **Model building**: We built three different models for sentiment analysis - LSTM, 1D CNN, and BERT.
6. **Training the models**: We trained each model on the tokenized and padded data using the Keras fit() function.
7. **Model evaluation**: We evaluated the performance of each model using accuracy, precision, recall, and F1 score. We also compared the performance of each model to identify the best model.
8. **Compare and Choose best performing Model**-: We compared the performance of LSTM, 1D CNN, and BERT models based on accuracy, precision, recall, and F1 score.

Data Loading

```
In [2]: df=pd.read_csv('226482609976817_File.csv')
```

```
In [3]: df.head(5)
```

Out[3]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	n
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	

Data preprocessing

We perform the following preprocessing steps on the tweets:

1. Remove user mentions, URLs, and special characters.
2. Convert all the text to lowercase.
3. Tokenize the text into words.
4. Remove stop words and words with length less than three characters.
5. Pad the sequences to a fixed length using the Keras padding function.

Models

- Here we are using below models and we will compare them:

1. LSTM
2. 1D-CNN
3. Pre-Trained BERT

LSTM

- LSTM is a type of RNN that works well for processing sequential data like text.
- The code defines an LSTM model using Keras for text classification.
- The model.summary() function provides a summary of the model architecture.
- LSTMs are powerful for text classification but may require more resources and training time than other models like 1D-CNNs or pre-trained language models like BERT.

TRAINING

- Here we have trained the model ,it took 10 epochs to complete the process and we got accuracy around 77%.

In [47]:
```
# Train the LSTM model
model.fit(xtrn, y_train, batch_size=64, epochs=10, validation_data=(xtst, y_test))

# Evaluate the LSTM model
loss, accuracy = model.evaluate(xtst, y_test)
print('Test accuracy:', accuracy)
```

```
Epoch 1/10
183/183 [==============================] - 6s 23ms/step - loss: 0.7805 - accuracy: 0.6719 - val_loss: 0.5996 - val_ac
curacy: 0.7544
Epoch 2/10
183/183 [==============================] - 3s 19ms/step - loss: 0.5223 - accuracy: 0.7894 - val_loss: 0.5429 - val_ac
curacy: 0.7927
Epoch 3/10
183/183 [==============================] - 3s 19ms/step - loss: 0.3792 - accuracy: 0.8622 - val_loss: 0.5490 - val_ac
curacy: 0.7937
Epoch 4/10
183/183 [==============================] - 3s 19ms/step - loss: 0.2893 - accuracy: 0.9001 - val_loss: 0.5925 - val_ac
curacy: 0.7705
Epoch 5/10
```

1D-CNN

- The model consists of an embedding layer, a 1D convolutional layer, a global max pooling layer, two fully connected layers, and a dropout layer.
- The embedding layer converts the input text data into a dense vector representation.
- The model is compiled using categorical cross-entropy loss and the Adam optimizer.
- The model summary provides information about the model architecture, including the number of parameters and the shapes of the input and output tensors.

Training

- Here we have trained the model ,it took 10 epochs to complete the process and we got accuracy around 76%.

In [51]:
```
# Train the LSTM model
model.fit(xtrn, y_train, batch_size=64, epochs=10, validation_data=(xtst, y_test))

# Evaluate the LSTM model
loss, accuracy = model.evaluate(xtst, y_test)
print('Test accuracy:', accuracy)
```

```
Epoch 1/10
183/183 [==============================] - 2s 8ms/step - loss: 0.8281 - accuracy: 0.6507 - val_loss: 0.6470 - val_acc
uracy: 0.7398
Epoch 2/10
183/183 [==============================] - 1s 7ms/step - loss: 0.5466 - accuracy: 0.7930 - val_loss: 0.5518 - val_acc
uracy: 0.7855
Epoch 3/10
183/183 [==============================] - 1s 7ms/step - loss: 0.3711 - accuracy: 0.8700 - val_loss: 0.5654 - val_acc
uracy: 0.7807
Epoch 4/10
183/183 [==============================] - 1s 6ms/step - loss: 0.2375 - accuracy: 0.9233 - val_loss: 0.6359 - val_acc
uracy: 0.7705
```

Pre-Trained BERT

- These uses the transformers library to load a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model called 'bert-base-uncased'.
- BERT is a state-of-the-art language model that can be fine-tuned for a variety of NLP tasks, including text classification.
- It also defines a function called preprocess_text_for_bert(), which takes a list of texts, a tokenizer object, and a maximum sequence length as inputs.
- The function tokenizes the texts using the tokenizer object, truncates or pads the sequences to the specified length, and returns two numpy arrays containing the input_ids and attention_masks for the BERT model.

Training

After we train these model, we got accuracy around 88%

In [69]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(
    [X_train, train_masks], y_train,
    validation_data=([X_val, val_masks], y_val),
    epochs=5,
    batch_size=32
)
```

```
Epoch 1/5
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.: `config=XConfig.from_pretrained('name',
output_attentions=True)`).
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0', 'tf_bert_model/b
ert/pooler/dense/bias:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `l
oss` argument?
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0', 'tf_bert_model/b
ert/pooler/dense/bias:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `l
oss` argument?
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.: `config=XConfig.from_pretrained('name',
output_attentions=True)`).
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0', 'tf_bert_model/b
ert/pooler/dense/bias:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `l
oss` argument?
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0', 'tf_bert_model/b
ert/pooler/dense/bias:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `l
oss` argument?
```

Results

- We evaluate the models on a test set, which contains 30% of the data. The following table shows the accuracy of each model:
- As we can see, BERT outperforms the other two models, achieving an accuracy of 88.5%.

Model	Accuracy
LSTM	79.6%
1D CNN	76.5%
BERT	88.5%

Conclusion

- In this project, we performed sentiment analysis on a dataset of airline tweets using three different models - LSTM, 1D CNN, and BERT. We achieved the highest accuracy using the BERT model, which shows the effectiveness of pre-trained language models in natural language processing tasks.