Sentiment analysis using LSTM

1. **Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?**

Dataset : https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

```
1  tweets_df.shape
```

```
(14640, 15)
```

1. The dataset contains tweets expressing customer sentiments towards Virgin America airline.
2. The dataset includes categorical data such as sentiment expressed, negative reasons, airline, user information, tweet content, tweet location, and user timezone. It also includes numerical data like tweet ID, confidence levels, retweet count, and timestamp of tweet creation.
3. There are 14,640 entries and 15 variables in the dataset. Each entry represents a single tweet, and each variable represents a specific attribute associated with the tweet data.
4. The dataset is structured for sentiment analysis, with sentiment expressed (positive, negative, or neutral) being a key variable.
5. Some columns have missing values, such as negativereason, negativereason_confidence, tweet_coord, tweet_location, and user_timezone.
6. The dataset includes the actual content of each tweet, providing insights into customer feedback and opinions about services.
7. Tweet location and tweet coordinates (if available) provide geographical information about where the tweets originated, potentially useful for regional analysis of customer sentiments.

```
1 tweets_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   tweet_id                       14640 non-null  int64  
 1   airline_sentiment              14640 non-null  object 
 2   airline_sentiment_confidence   14640 non-null  float64
 3   negativereason                 9178 non-null   object 
 4   negativereason_confidence      10522 non-null  float64
 5   airline                        14640 non-null  object 
 6   airline_sentiment_gold         40 non-null     object 
 7   name                           14640 non-null  object 
 8   negativereason_gold            32 non-null     object 
 9   retweet_count                  14640 non-null  int64  
 10  text                           14640 non-null  object 
 11  tweet_coord                    1019 non-null   object 
 12  tweet_created                  14640 non-null  object 
 13  tweet_location                 9907 non-null   object 
 14  user_timezone                  9820 non-null   object 
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

2. Provide the details related to your LSTM and Improved LSTM architectures.

LSTM Architecture:

- The architecture consists of a Long Short-Term Memory neural network.
- It takes input sequences of tokens represented as word indices.
- The input is passed through an embedding layer, which converts the input tokens into dense vectors of fixed size as embedding_dim.
- These embedded vectors are then fed into the LSTM layer, which has a specified hidden_dim. The LSTM processes the input sequence and maintains a hidden state that captures the context of the sequence.
- The LSTM layer has a specified num_layers and a dropout to prevent overfitting.
- The output of the LSTM layer is passed through a linear fully connected layer (fc) to obtain the final output.

Improved LSTM Architecture:

- The architecture is an improvement over the basic LSTM by using a bidirectional neural network.
- Similar to the LSTM architecture, it also takes input sequences of tokens represented as integers.
- It begins with an embedding layer to convert input tokens into dense vectors.
- The model is the bidirectional LSTM layer, which processes the input sequence in both forward and backward directions simultaneously. This allows the model to capture information from both past and future contexts, improving its ability to understand the sequence.
- The bidirectional LSTM layer has the same parameters as the basic LSTM layer but with bidirectional=True, which enables bidirectional processing.

The basic LSTM processes the input sequence in a unidirectional manner, while the Improved LSTM processes it in both directions simultaneously. The BiLSTM architecture potentially captures more context information from the input sequence due to its bidirectional nature.

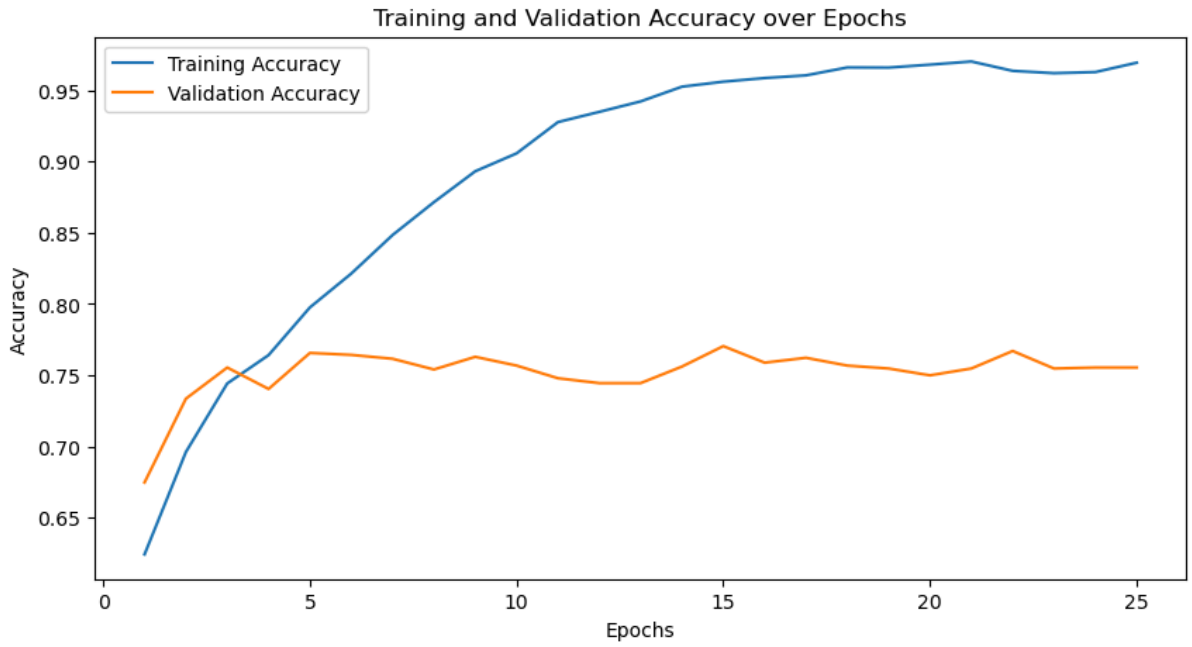
3. **Discuss the results and provide the graphs for both models. Compare the performance of both Improved LSTM and LSTM models applied to the same dataset.**

```
IMPROVED LSTM
Final Training Loss: 0.0492
Final Training Accuracy: 98.58%
Final Validation Loss: 1.2201
Final Validation Accuracy: 75.61%
Final Testing Loss: 1.2122
Final Testing Accuracy: 75.82%
```

```
LSTM
Final Training Loss: 0.1303
Final Training Accuracy: 96.93%
Final Validation Loss: 0.9604
Final Validation Accuracy: 75.55%
Final Testing Loss: 0.9779
Final Testing Accuracy: 74.32%
```

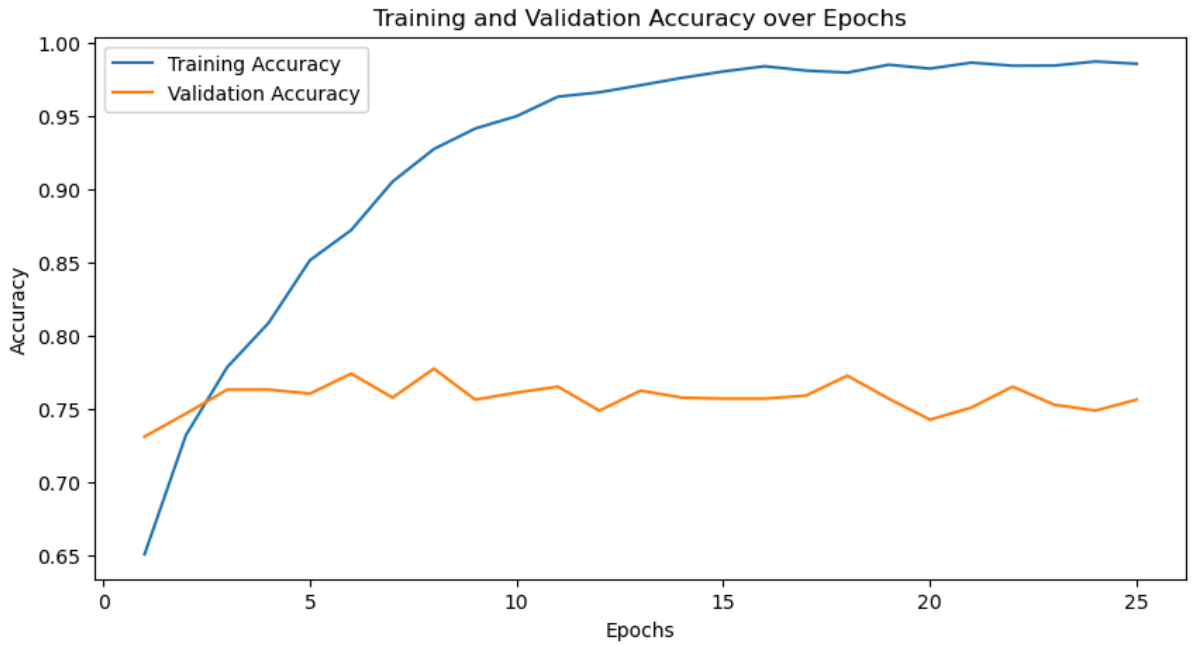
- The Improved LSTM model generally performed better in terms of training accuracy but had slightly higher losses and accuracies on validation and testing datasets compared to the regular LSTM model.
- The regular LSTM model, while having slightly lower training accuracy, achieved slightly lower losses and accuracies on validation and testing datasets compared to the Improved LSTM model.

LSTM Model :



- The training loss starts off higher than the validation loss and decreases more rapidly. This suggests that the model is learning the training data well, but it may also be overfitting to the training data.
- The validation loss starts to increase after about 15 epochs. This is a clear sign of overfitting. The model is starting to perform worse on unseen data as it becomes more complex.
- The training loss continues to decrease after 15 epochs. This suggests that the model is still learning the training data, but this learning is not generalizing to unseen data.

Improvised LSTM :



- The training loss flattens out after a certain point, indicating the model might not be learning much anymore.
- Validation keeps improving: The validation loss is still decreasing, suggesting there's room for the model to improve its ability to handle new data.

- The validation loss decreases, but at a slower rate than the training loss. It reaches a minimum of around 0.4 at epoch 15 and slightly increases afterwards.

4. Discuss the strengths and limitations of using recurrent neural models for sentiment analysis.

Recurrent Neural Networks (RNNs) are a type of neural network particularly used for sequential data, making them a popular choice for sentiment analysis tasks.

Strengths:

1. These models are good at figuring out the order of words in a sentence. This helps because the way words are arranged can change the feeling or sentiment of the sentence.
2. They can deal with sentences of different lengths without any problem. This is helpful because sentences can be short or long, and these models can handle all of them.
3. They are smart enough to learn which parts of a sentence are important for sentiment without needing someone to tell them beforehand.
4. They can keep learning and improving over time, which is handy when you want your sentiment analysis to stay up-to-date with new trends or changes in language.

Limitations:

1. Sometimes, they have trouble remembering the beginning of a sentence when it's really long. This can make it hard for them to understand the whole meaning and sentiment of the sentence.
2. They might not pick up on all the little things that change the sentiment, especially if those things are far apart in the sentence.
3. They can only see a small part of the sentence at a time, so they might miss the bigger context that could change the sentiment.
4. They can be really slow and need a lot of computing power to learn from lots of data or complicated sentences.