User Sentiment Analysis using LSTM

A Multi-Domain Application in Business Analysis, Cybersecurity, and Customer Support

Ajmal M S

August 19, 2025

Abstract

This project was built, keeping business analysis, cybersecurity, and customer insights in mind to leverage the benefit of user reviews analysis using Data Analytics methods and tools. I started the base of this project at my grad school project, back in 2022. From that time, I have been improvising the project and analyzing applications to establish a meaning to this wonderful technology. Think about this: you are a social media influencer. You are selling the most hyped beauty product. Due to the presence you have on social media platforms, you are getting tens of thousands of comments every day, which obviously neither you nor your employees can read and understand valuable feedback. This is the time when my idea helps to analyze all your user comments in one go and segregate them as per the sentiment (negative/positive) of the user. This project leverages Natural Language Processing (NLP) and Deep Learning (LSTM) to analyze user sentiment from an airline customer reviews data set. Beyond technical implementation, this work demonstrates how sentiment analysis can be applied across three professional domains:

1. Business Analysis for deriving customer insights
2. Cybersecurity for monitoring brand reputation and emerging threats
3. Customer Support for ticket categorization and service improvement.

.

Contents

1 Introduction **2**

1.1 Background 2

1.2 Objectives 3

2 Dataset and Preprocessing **4**

2.1 Dataset 4

2.2 Pre-processing Steps 4

2.2.1 Removing neutral labels 4

2.2.2 Tokenization 5

2.2.3 Sequence padding 5

2.2.4 Why I Removed Neutral 5

2.3 Code Example 6

3 Model Development **7**

3.1 Architecture 7

3.2 Code Example 7

3.3 Training 8

3.4 Results 8

4 Use Cases **10**

4.1 Business Analyst Perspective 10

4.2 Cybersecurity Perspective 10

4.3 Customer Support Perspective 11

4.4 Streamlit Interface for Sentiment Analysis 11

5 Conclusion and Future Work **12**

5.1 Conclusion 12

5.2 Future Work 12

A Full Code Listing **14**

Chapter 1

Introduction

1.1 Background

Data abundance in every industry has been increasing since the digitization of the 2010s. This data abundance is simply the user reviews in the comment section in a text format. So, since the comment/review data is huge, nobody can sit and read all the comments to earn Feedback about a particular product. Sentiment Analysis (SA) is used for determining the positive, neutral, and negative comments based on Natural Language Processing (NLP).

- If a customer/user posts a comment like "This will be Apple's best phone", the "good" comment will be classified as a positive comment and the organization will improve the production in that domain.
- If a customer/user posts a comment like "Apple's worst phone in the software segment", the "worst" comment will be classified as a negative comment, and the organization will rectify the product quality.

Opinions are declarations that express a person's perception or emotion. Sentiment analysis is a set of methodologies, strategies, and tools for identifying and extracting subjective information from language, such as opinions and attitudes [1], which aids in determining the customers' sentiments toward buying a specific product or issue. As already stated, data abundance in online business has been increasing since digitization from yearly 2010s, especially product reviews in text format. Since the comment/review data is huge, reading all the comments to earn feedback about a particular product is far from a possibility. Solution for this is Sentiment Analysis. Let's see a classic example of an in-store business service review:

- If a customer/user post a comment like "The pizza is really good here, but the service was slow," the first half of the comment will be classified as a positive comment using NLP, and the business can leverage it as its unique selling point.

- But, the second half of the comment will be classified as a negative comment using NLP, and the business will get an insight into how to improve the service. Here it is, the "service time".

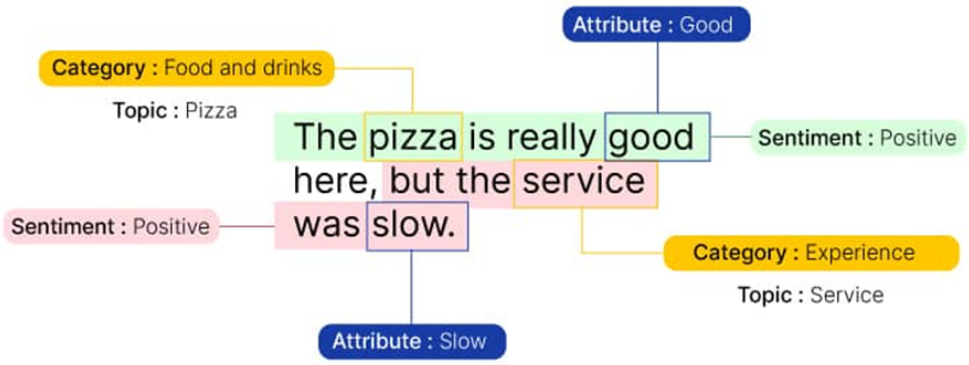

Figure 1.1: Example of an in-store business service review

1.2 Objectives

Of all the internet users worldwide, those who were using social media actively are more than 63.9 percent [4]. People nowadays are ready to spend their quality amount of time in social media and online channels like e-commerce platforms. All the feedbacks about a change, a product, a service, or an incident are shared on social media fluently. As an aspiring **Business Analyst** and a concerned **Cybersecurity** Enthusiast, as well as an experienced **Customer Services** specialist, my objective for this project is to transform abundant, unstructured feedback into insight-rich, sentiment-scored information, which can help online businesses to grow, adapt, and respond to the feedback cycles (addressing potential cybersecurity attacks too). My workflow is as follows:
Choosing a credible dataset (like this one from Kaggle) >Using a good Programming Language¿Importing necessary libraries >Cleaning and analysing data >Declaring models >Operating >Testing >Predicting Sentiment (Positive/Negative/Neutral) of a comment

Chapter 2

Dataset and Preprocessing

2.1 Dataset

I used the Twitter US Airline Sentiment data set from Kaggle. This dataset helps analyze how travelers in February 2015 expressed their feelings on Twitter. This data originally came from Crowdflower's Data for Everyone library. As the original source says, "A sentiment analysis job about the problems of each major U.S. airline. Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as "late flight" or "rude service")." [2]. The collaborators of this data set are - Figure Eight (Owner), and Kaggle Team Bot (Admin). It holds a CC BY-NC-SA 4.0 license. With Not specified update Frequency(Updated 6 years ago).

2.2 Pre-processing Steps

This data set is complex for a data analysis/predictive model. It contains 15 columns with categorical variables, numerical variables, binary variables, and time variables. In order to prepare the dataset for training a deep learning model, several pre-processing steps were applied. These steps ensure that the raw text data is transformed into a numerical format that the model can understand.

2.2.1 Removing neutral labels

The original dataset contained three classes of sentiment: *positive*, *negative*, and *neutral*. Since the goal of this project was to build a binary classification model focusing only on clear sentiment polarity, the neutral class was removed. This step not only simplified the classification task but also helped improve the clarity of results by avoiding ambiguity.

```
tweet_df_filtered = tweet_df[tweet_df['airline_sentiment'] != '
    neutral']
```

```
sentiment_label, unique = tweet_df_filtered.airline_sentiment.
    factorize()
```

2.2.2 Tokenization

Tokenization is an important feature to divide a sentence into smaller units (words/letters), so later in the stage, we can apply padding (numbers), which eases analysis or processing at the word level in natural language processing tasks. Think about it, a human can understand a language or a dialogue from a fellow being. We catch the meaning of the sentence by analyzing the sequence or grammar (basically a rule). But in a computer, the context of natural language is understood word-for-word (just like 1 = ON and 0 - OFF) [3]. **Deep learning models cannot directly interpret raw text. Therefore, each review (tweet) was converted into sequences of integers using a tokenizer. The tokenizer assigns a unique index to each word based on its frequency in the dataset. In this project, the vocabulary size was restricted to 5000 most frequent words, ensuring that rare words with limited impact did not introduce noise.**

```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(tweet)
encoded_docs = tokenizer.texts_to_sequences(tweet)
```

2.2.3 Sequence padding

Since reviews vary in length, padding was applied to standardize all input sequences to the same length. This is necessary for batch processing in neural networks, which require inputs of equal size. In this case, each review was padded (or truncated) to 200 tokens, ensuring consistency while preserving sufficient context.

```
padded_sequence = pad_sequences(encoded_docs, maxlen=200)
```

After these preprocessing steps, the text data was successfully transformed into numerical input suitable for training an LSTM model.

2.2.4 Why I Removed Neutral

In this project, neutral labels were excluded during preprocessing. The rationale is that neutral statements often do not carry actionable insights for business decision-making, cybersecurity monitoring, or customer support automation. For example, “I am at the airport” does not provide any positive or negative sentiment, making it less valuable in generating KPIs, detecting risks, or triggering escalation workflows.

By removing neutral data, the model focuses solely on distinguishing between positive and negative opinions, which directly impact business outcomes. This binary classification is more efficient and interpretable in real-world applications.

2.3 Code Example

The following code snippet demonstrates the preprocessing steps applied to the dataset before training the model:

```
tweet_df_filtered = tweet_df[tweet_df['airline_sentiment'] != '
    neutral']
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(tweet)
padded_sequence = pad_sequences(encoded_docs, maxlen=200)
```

- **Line 1:** `tweet_df_filtered = tweet_df[tweet_df['airline_sentiment'] != 'neutral']` This line removes all rows where the sentiment label is *neutral*. By doing so, the dataset now only contains positive and negative reviews, making the problem a binary classification task.
- **Line 2:** `tokenizer = Tokenizer(num_words=5000)` A tokenizer object is created, which will later convert text into numbers. The parameter `num_words=5000` ensures that only the 5000 most frequent words in the dataset are considered, while ignoring rare words.
- **Line 3:** `tokenizer.fit_on_texts(tweet)` The tokenizer is fitted on the entire text dataset. This means it scans all reviews, builds a vocabulary, and assigns a unique index (integer) to each word based on its frequency.
- **Line 4:** `padded_sequence = pad_sequences(encoded_docs, maxlen=200)` After converting words into sequences of integers, the lengths of the reviews are not uniform. To make them consistent, padding is applied. Here, all reviews are either truncated or padded with zeros so that every review has exactly 200 tokens. This is required because neural networks expect inputs of the same size.

In summary, this preprocessing ensures the dataset is clean, numerical, and standardized, making it suitable for feeding into a deep learning model.

Chapter 3

Model Development

3.1 Architecture

The model used in this project is a deep learning architecture built with the Keras library. It is designed to process text data and classify reviews into positive or negative sentiments. The main components are:

- **Embedding Layer:** Converts words into dense vector representations. Each word is mapped to a 32-dimensional vector that captures semantic meaning. This allows the model to understand relationships between words instead of treating them as independent tokens.
- **LSTM Layer:** A Long Short-Term Memory (LSTM) network is used to capture dependencies in sequential data. Since customer reviews are sentences with context, LSTMs are ideal for remembering important words over longer sequences.
- **Dropout Layers:** Dropout is applied to reduce overfitting. By randomly "dropping" units during training, the network becomes more generalizable. A spatial dropout is used after the embedding layer, and standard dropout is applied within the LSTM.
- **Dense Output Layer:** A fully connected dense layer with a sigmoid activation function outputs the probability of a review being positive or negative. Since this is a binary classification problem, the sigmoid activation is appropriate.

3.2 Code Example

```
The following code snippet shows the construction of the model
   architecture:

\begin{lstlisting}
```

```
model = Sequential()
model.add(Embedding(vocab_size, 32, input_length=200))
model.add(SpatialDropout1D(0.25))
model.add(LSTM(50, dropout=0.5, recurrent_dropout=0.5))
model.add(Dense(1, activation='sigmoid'))
```

- **Embedding:** Maps each word into a 32-dimensional vector space. The parameter `input_length=200` ensures that all input sequences are of uniform length.
- **SpatialDropout1D(0.25):** Drops 25% of embedding dimensions at random, which improves robustness of the model.
- **LSTM(50):** The LSTM layer has 50 units (neurons). Dropout values of 0.5 are applied both to the input and recurrent connections, preventing overfitting.
- **Dense(1, sigmoid):** A single output neuron produces a value between 0 and 1, representing the probability of positive sentiment.

3.3 Training

During training, the following strategies were applied:

- **Validation Split:** 20% of the dataset was held out for validation to measure model performance on unseen data.
- **Early Stopping:** Training was stopped early if the validation loss did not improve for 3 consecutive epochs. This prevented overfitting and reduced unnecessary computation.
- **Class Weights:** Since the dataset had an imbalance between positive and negative samples, class weights were applied. This ensured that the model did not become biased towards the majority class.

3.4 Results

The model achieved good accuracy on the validation set. Figure 3.1 shows the accuracy and loss over epochs:

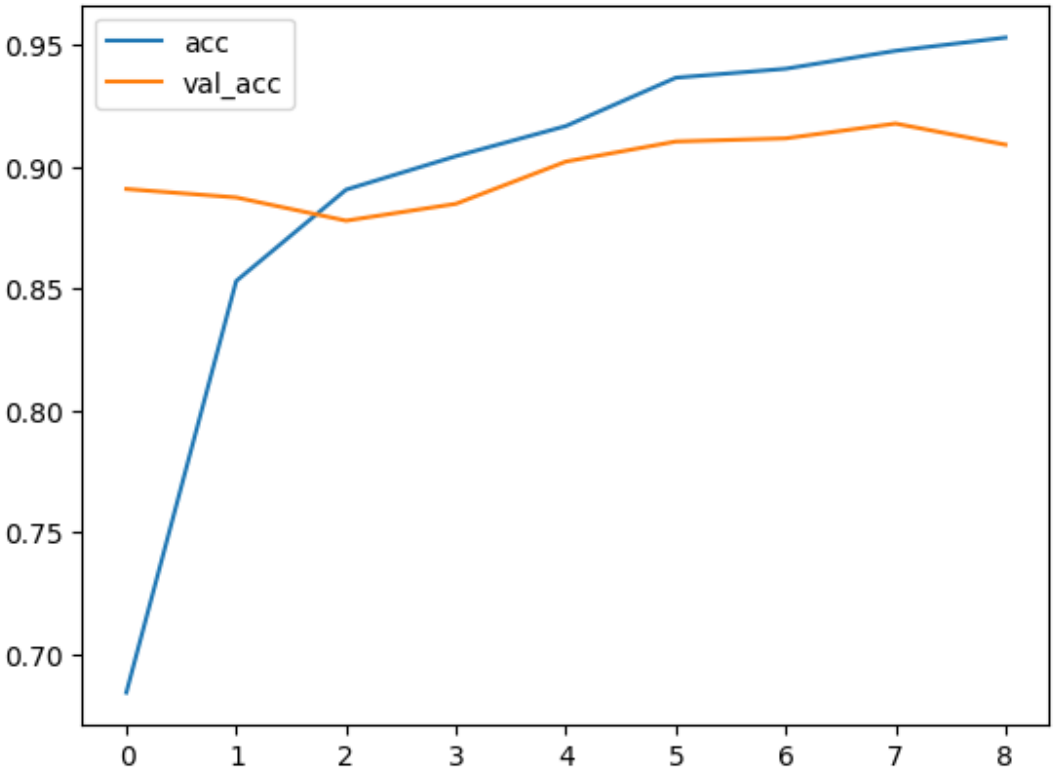

Figure 3.1: Model Accuracy over Epochs

The model predicted the sentence ”I enjoyed my journey on this flight.” as **positive** and the sentence ”This is the one bad flight experience of my life!” as **negative**. The output visualization is shown in Figure 4.1.

```
test_sentence1 = "I enjoyed my journey on this flight."
predict_sentiment(test_sentence1)
test_sentence2 = "This is the one bad flight experience of my life!"
predict_sentiment(test_sentence2)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 412ms/step
Predicted label: positive (probability: 0.108)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
Predicted label: negative (probability: 0.868)
('negative', 0.8676613569259644)
```

Figure 3.2: Reviews Test

Chapter 4

Use Cases

4.1 Business Analyst Perspective

- From a business analyst's point of view, sentiment analysis provides a powerful way to transform raw customer feedback into actionable insights. By analyzing the distribution of positive and negative sentiments, organizations can track key performance indicators (KPIs) such as overall customer satisfaction, Net Promoter Score (NPS), and brand perception.
- These insights can be visualized in dashboards to monitor trends over time. For example, a sudden spike in negative sentiment after a product launch could highlight quality issues or gaps in customer expectations. Based on these patterns, analysts can recommend targeted actions such as product improvements, staff training, or marketing adjustments.
- In short, sentiment results help business analysts bridge customer voice with data-driven decisions that directly impact strategy and performance.

4.2 Cybersecurity Perspective

- Although sentiment analysis is most commonly used for marketing and customer experience, it also provides value in cybersecurity monitoring. Negative sentiment in customer reviews or social media posts may include mentions of fraud, data breaches, phishing attempts, or security concerns.
- For instance, if multiple customers post negative feedback about suspicious transactions or account issues, these can serve as early warning signals for potential security risks. By flagging such comments, cybersecurity teams can prioritize investigations and correlate with incident reports.

- Thus, sentiment analysis can complement threat intelligence by capturing human-reported risks, enabling faster detection and response to potential security threats.

4.3 Customer Support Perspective

- In customer support, automation through sentiment analysis improves efficiency and responsiveness. Incoming tickets, emails, or chat messages can be automatically classified based on sentiment. Negative sentiment cases can be escalated to senior support staff for immediate attention, while positive or neutral cases can be routed to standard workflows.
- Furthermore, analyzing sentiment across large volumes of support requests helps identify common complaints and recurring pain points. For example, if many negative reviews mention ”payment issues,” this insight can be escalated to the product or finance teams for resolution.
- Overall, sentiment analysis empowers support teams to prioritize cases, reduce resolution time, and enhance customer satisfaction by ensuring that the most critical issues are addressed first.

4.4 Streamlit Interface for Sentiment Analysis

A simple Streamlit interface allows customer support teams to quickly input user reviews or comments and receive instant sentiment predictions. This visual and interactive tool helps support staff identify negative feedback early, prioritize responses, and make data-driven decisions to improve customer satisfaction.

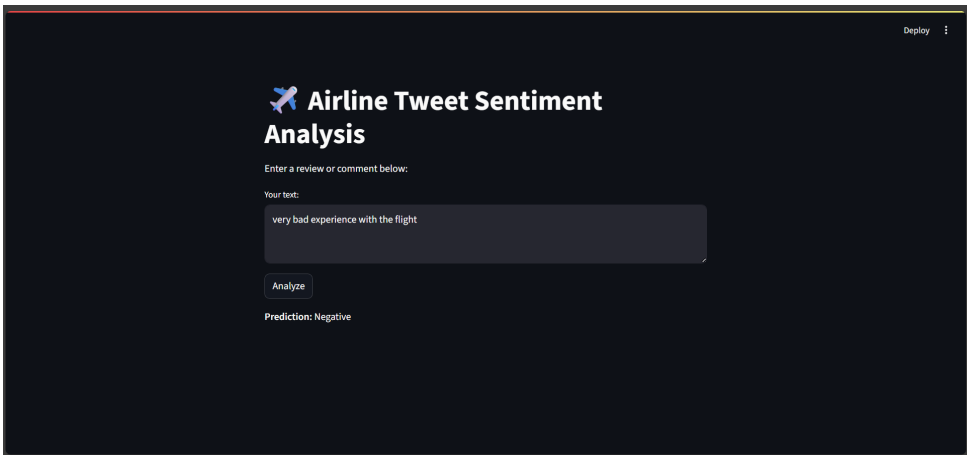

Figure 4.1: Streamlit interface

Chapter 5

Conclusion and Future Work

5.1 Conclusion

- The LSTM-based sentiment analysis model achieved strong performance on the airline tweet dataset. Training accuracy steadily improved from 53.9% in the first epoch to 95.9% by the ninth epoch. More importantly, the model generalized well to unseen data, with validation accuracy stabilizing in the range of 88–91%, and peaking at 91.7%. This indicates that the model effectively learned patterns of positive and negative sentiment while avoiding significant overfitting.
- he results validate the suitability of deep learning for real-world text classification tasks. From a cross-domain perspective, the project illustrates versatility:
  - **Business analysts** can convert sentiment outcomes into KPIs, dashboards, and actionable insights.
  - **Cybersecurity professionals** can monitor negative sentiment for signals of fraud, scams, or emerging risks.
  - **Customer support teams** can automate ticket triaging, escalation, and trend detection.

5.2 Future Work

- Despite the promising results, several limitations remain. The dataset used is approximately six years old, meaning it may not fully represent current modes of digital expression. In recent years, Gen-Z vocabulary, internet slang, and the heavy use of emojis have become central to online communication, yet these are underrepresented in the dataset. This restricts the model's ability to interpret more contemporary forms of sentiment.
  - Expanding and updating the dataset with modern reviews, tweets, and customer feedback that incorporate emojis, slang, and other emerging forms of expression.

- Experimenting with transformer-based models (e.g., BERT, RoBERTa, or GPT variants), which are more context-aware and robust for non-standard language.
- Deploying the model in interactive applications using Streamlit or Gradio, enabling real-time analysis of user-provided text.
- Exploring multilingual sentiment analysis to extend applicability across regions and diverse customer bases.

In summary, the current model demonstrates accuracy above 91% on validation data and proves adaptable across business, security, and customer-facing domains. By updating datasets and adopting modern architectures, the system can remain relevant and impactful in contemporary applications.

Appendix A

Full Code Listing

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense, Dropout,
   SpatialDropout1D
from tensorflow.keras.layers import Embedding

from google.colab import drive
drive.mount('/content/drive')

df=pd.read_csv('/content/drive/MyDrive/User Sentiment Analysis/
   Tweets Data Set.csv')

df.head()

df.columns

tweet_df = df[['text','airline_sentiment']]
print(tweet_df.shape)
tweet_df.head(5)

tweet_df['airline_sentiment'].value_counts().plot(kind='bar')
plt.title("Sentiment Distribution")
```

```
plt.show()

print(tweet_df.shape)

tweet_df.head(5)

tweet_df["airline_sentiment"].value_counts()

# After filtering neutrals:
tweet_df_filtered = tweet_df[tweet_df['airline_sentiment'] != '
    neutral']

# Then factorize on the filtered DataFrame:
sentiment_label, unique = tweet_df_filtered.airline_sentiment.
    factorize()
tweet = tweet_df_filtered.text.values
print(sentiment_label.shape)  # Should be (11541,)
print(unique)  # Should only include 'positive' and 'negative'

tweet = tweet_df_filtered.text.values

tokenizer = Tokenizer(num_words=5000)

tokenizer.fit_on_texts(tweet)

vocab_size = len(tokenizer.word_index) + 1

encoded_docs = tokenizer.texts_to_sequences(tweet)

padded_sequence = pad_sequences(encoded_docs, maxlen=200)

print(tokenizer.word_index)

print(tweet[0])

print(encoded_docs[0])

print(padded_sequence[0])

embedding_vector_length = 32
```

```
model = Sequential()

model.add(Embedding(vocab_size, embedding_vector_length,
   input_length=200))

model.add(SpatialDropout1D(0.25))

model.add(LSTM(50, dropout=0.5, recurrent_dropout=0.5))

model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam', metrics=[
   'accuracy'])

print(model.summary())

from tensorflow.keras.callbacks import EarlyStopping
from sklearn.utils import class_weight

early_stop = EarlyStopping(monitor='val_loss', patience=3,
   restore_best_weights=True)
class_weights = class_weight.compute_class_weight('balanced',
   classes=np.unique(sentiment_label), y=sentiment_label)
class_weights_dict = dict(enumerate(class_weights))

# Train the model with early stopping and class weights
history = model.fit(padded_sequence, sentiment_label,
   validation_split=0.2,
                    epochs=10, batch_size=32, class_weight=
                       class_weights_dict,
                    callbacks=[early_stop])

# Save the trained model
model.save('sentiment_model.h5')

# Save the tokenizer for later use
import pickle
with open('tokenizer.pkl', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

```
plt.plot(history.history['accuracy'], label='acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.legend()
plt.show()
plt.savefig("Accuracy plot.jpg")

plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
plt.savefig("Loss plot.jpg")

def predict_sentiment(text):
    tw = tokenizer.texts_to_sequences([text])
    tw = pad_sequences(tw, maxlen=200)
    prob = model.predict(tw).item()
    label = unique[int(round(prob))]
    print(f"Predicted label: {label} (probability: {prob:.3f})")
    return label, prob

test_sentence1 = "I enjoyed my journey on this flight."
predict_sentiment(test_sentence1)
test_sentence2 = "This is the one bad flight experience of my life!"
predict_sentiment(test_sentence2)
```

References

[1] CLARIN. Social media users. https://www.clarin.eu/resource-families/tools-sentiment-analysis-and-opinion-mining#:~:text=Tools%20for%20sentiment%20analysis%20and%20opinion%20mining%20in,et%20%20...%20%201%20more%20rows%20, 2014.

[2] Crowdflower. Twitter us airline sentiment dataset. https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment/data, 2015.

[3] Utkarsh Kant. Tokenization. https://medium.com/@utkarsh.kant/tokenization-a-complete-guide-3f2dd56c0682, 2022.

[4] Ani Petrosyan. Social media users. https://www.statista.com/statistics/617136/digital-population-worldwide/#:~:text=As%20of%20February%202025%2C%205.56%20billion%20individuals%20worldwide,of%20the%20world%27s%20population%2C%20were%20social%20media%20users., 2025.