Sentiment and Emotion Classification on Movie Bird Box (2018) Reviews

Presented By:

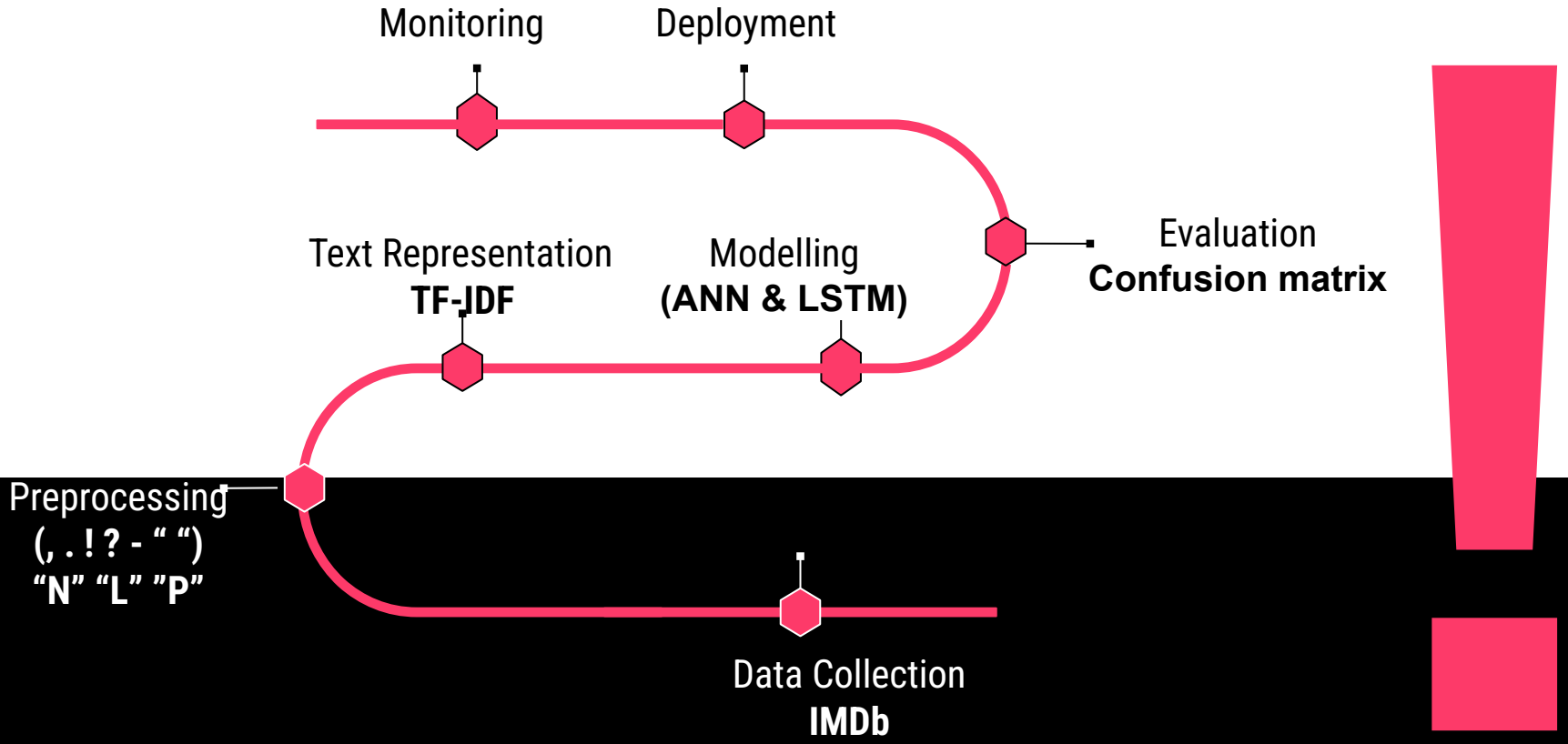
Hanaa Mahmoud Nur :

Duaa Omar ALZahrani :

Ghada Abdullah ALsulami :

Raseel ALghamdi :

Table of Contents



Classical ML Pipeline

Introduction

Nowadays , people tend to be opened and express their opinion about what is going around ,not just by rating product or like it or dislike it in brief way .No, they could write up to 100 words or more , analys, criticize, express feeling like (Joy,Anger,Sadness,disappointment ,ect...) with social media , everything spread fast just by one Click .
This amount of world will take long time to be analyse , understand ,to get valuable insight . With ML,NLP pipeline process this task tend to easy

In this Project,

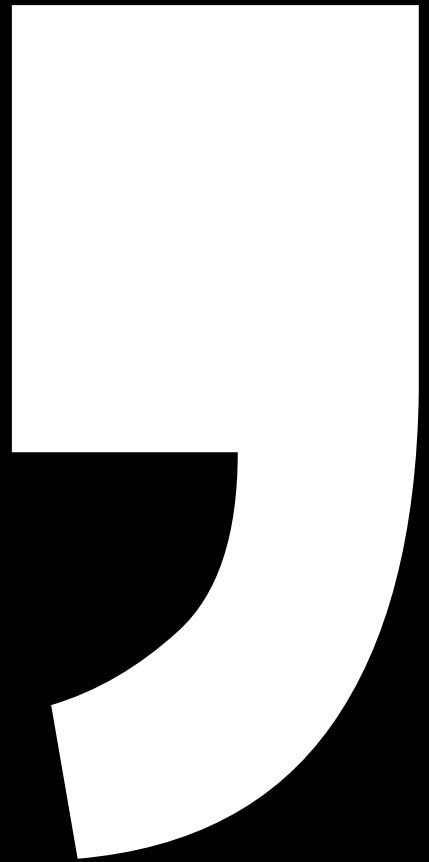
we chose to apply ML Pipeline on Bird Box movie reviews because it is an example of a movie that divided public opinion.
While some viewers found it emotionally intense and thrilling, others felt it didn't meet their expectations.

Objective

* TO analyse sentiment & emotional tone in movie reviews.

*Implementing ML Pipeline in NLP on real world problem .

*TO understand public opinion and emotional response on the movie.



Case study on:

Bird Box Movie Reviews (2018)

Data Collection

Data Source

*IMDb Movie Reviews Dataset .

*IEEE Dataport .

Data Retrieval

*Downloading a CSV file from IMDb

Data Overview

*The Dataset contains more than 2000 records

* We Extract about 800

Data Annotation

Detecting Sentiment & Emotion of user Reviews & Label

Sentiment

Label each as

+ −

Emotion

Label each as

😄 😠 😞

Preprocessing

1-Cleaning Reviews :

```python
# Preprocess Text

# Function to clean review text
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'[^a-z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

# Apply the cleaning function to the 'review' column
df['cleaned'] = df['review'].apply(clean_text)
```

	review	Sentiment	Label (emotion)	cleaned
0	I have to say, the most sensible reviews are t...	negative	Sadness	i have to say the most sensible reviews are th...
1	The movie was OK. Definitely better than the H...	Negative	Sadness	the movie was ok definitely better than the ha...
2	I have no complaints about the acting, product...	negative	Anger	i have no complaints about the acting producti...
3	I really can't stand loose ends. So many quest...	Negative	Anger	i really cant stand loose ends so many questio...
4	There are some fantastic short films out there...	negative	Anger	there are some fantastic short films out there...
5	Why. Why. Why. Why. Just why? Why do ...	Negative	Anger	why why why why just whybrbrwhy do people find...
6	If you have seen The Happening, you have LITER...	negative	Anger	if you have seen the happening you have litera...

Preprocessing

2-Cleaning Label :

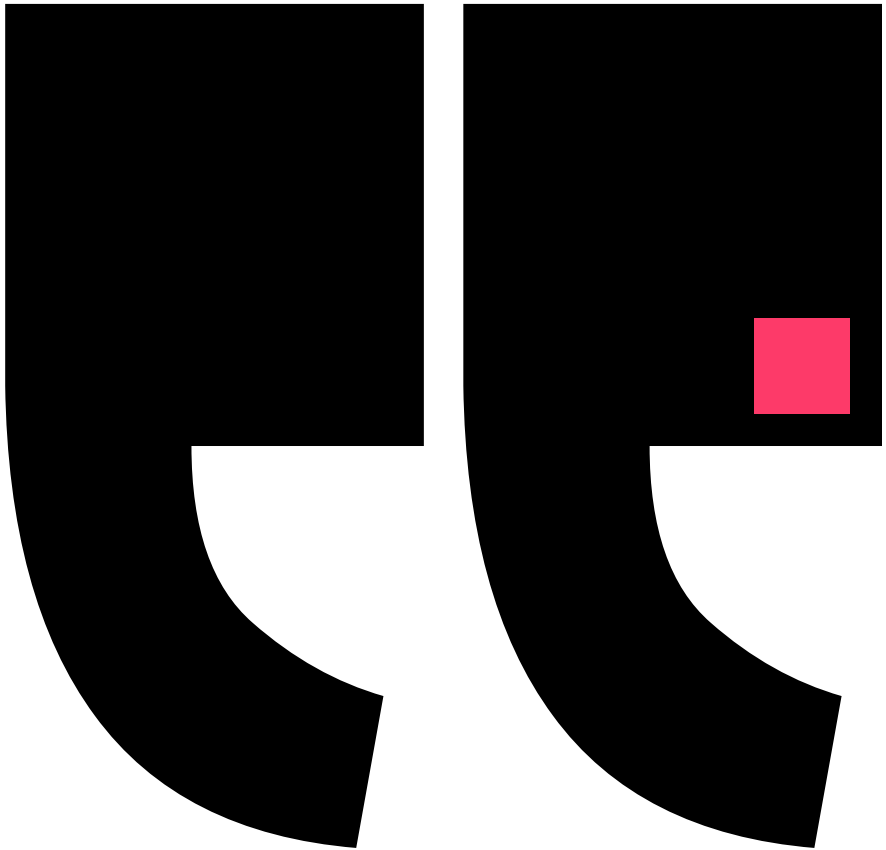
```
# clean the Lables
# Remove extra spaces from column names
df.columns = df.columns.str.strip()

# Clean the text in the columns
df['Sentiment'] = df['Sentiment'].apply(clean_text)
df['Label (emotion)'] = df['Label (emotion)'].apply(clean_text)

# Display the first 5 rows
display(df[['Sentiment', 'Label (emotion)']].iloc[:5])
```

	Sentiment	Label (emotion)
0	negative	sadness
1	negative	sadness
2	negative	anger
3	negative	anger
4	negative	anger

Text Preprocessing

3-Tokenization

```
#Text Representation using Tokenizer

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
tokenizer.fit_on_texts(df['clean_text'])

sequences = tokenizer.texts_to_sequences(df['clean_text'])
padded_sequences = pad_sequences(sequences, padding='post', maxlen=100)
```



Sentiment :

➖ ➡ 0, ➕ ➡ 1

Emotion:

😠 ➡ 0, 😄 ➡ 1, 😒 ➡ 2

Label Encoding

Text Representation

We Use **TF-IDF** to generate vectors representing textual data in order to be used as an input in ML models

```
Shape of TF-IDF matrix: (811, 5000)
Training set shape: (648, 5000)
Test set shape: (163, 5000)
```

ANN & LSTM

ANN:

Input: TF-IDF features

Layers:

-Dense → ReLU

-Dropout

-Output: Softmax

Simpler model, faster training

LSTM:

Input: Tokenized padded sequences

Layers:

-Embedding

-LSTM

-Dropout

-Output: Softmax

Captures word order/context

ANN - Sentiment Results

Training Accuracy:

100%

Validation Accuracy:

89%

Confusion Matrix:

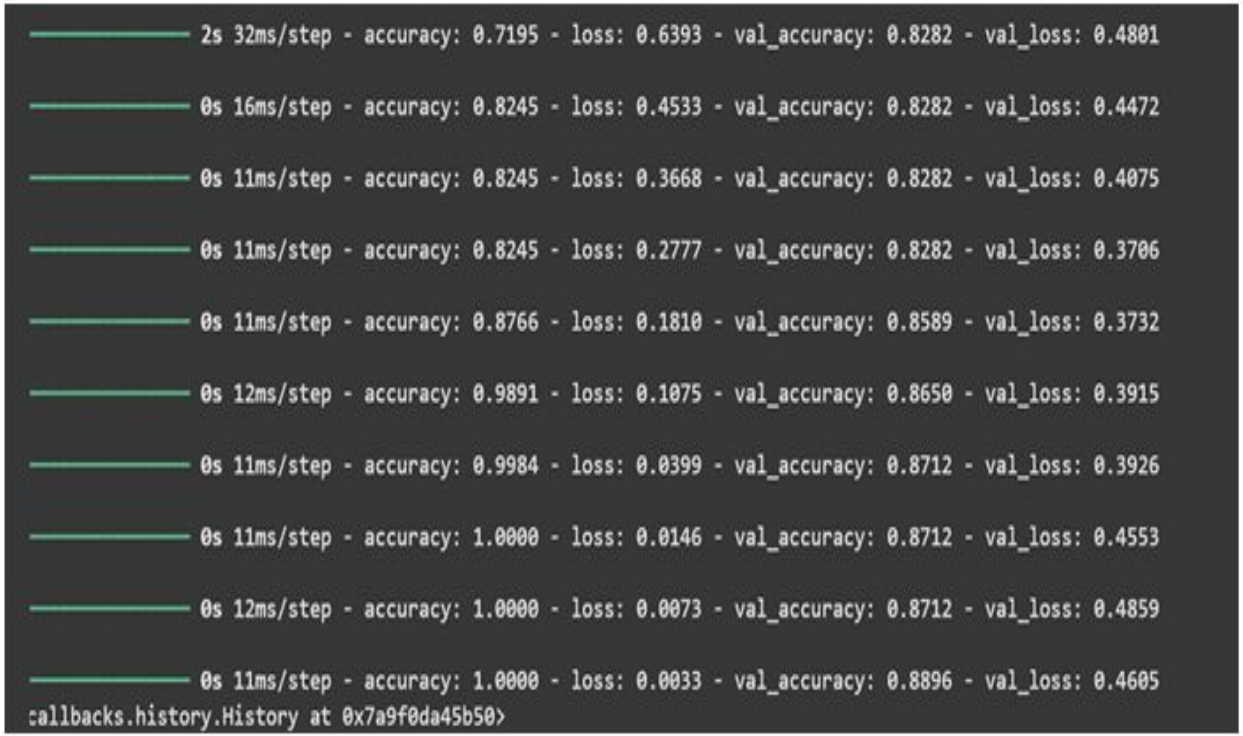
Perfect on negative samples
Missed 18/28 positive samples

F1-Score:

Negative: 0.94

Positive: 0.53

ANN - Sentiment Results



Training Results of the Sentiment Classification Model (ANN)

ANN - Emotion Results

Training Accuracy:

100%

Validation Accuracy:

67%

Performs:

Best at Anger,
struggles with **Sadness**

F1-Score:

Anger: 0.79

Joy: 0.48

Sadness: 0.00

ANN - Emotion Results

```
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - accuracy: 0.6180 - loss: 1.0475 - val_accuracy: 0.6319 - val_loss: 0.9244
Epoch 2/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6481 - loss: 0.8537 - val_accuracy: 0.6319 - val_loss: 0.8847
Epoch 3/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6481 - loss: 0.7485 - val_accuracy: 0.6319 - val_loss: 0.8341
Epoch 4/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6788 - loss: 0.6019 - val_accuracy: 0.6748 - val_loss: 0.7764
Epoch 5/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8442 - loss: 0.4422 - val_accuracy: 0.6933 - val_loss: 0.7562
Epoch 6/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.9565 - loss: 0.2721 - val_accuracy: 0.6687 - val_loss: 0.7973
Epoch 7/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9909 - loss: 0.1386 - val_accuracy: 0.6626 - val_loss: 0.9546
Epoch 8/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 1.0000 - loss: 0.0577 - val_accuracy: 0.6810 - val_loss: 0.9788
Epoch 9/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.0268 - val_accuracy: 0.6626 - val_loss: 1.0187
Epoch 10/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.0151 - val_accuracy: 0.6687 - val_loss: 1.0987
<keras.src.callbacks.history.History at 0x7a9f0dc12f10>
```

Training Results of the Emotion Classification Model (ANN)

LSTM - Sentiment Results

Training Accuracy:

88%

Validation Accuracy:

83%

Confusion Matrix:

134/135 negative correctly

0/28 positive correctly

F1-Score:

Negative: 0.90

Positive: 0.00

LSTM - Sentiment Results

```
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
21/21 ━━━━━━━━━━━━━━━━━━━━ 58s 159ms/step - accuracy: 0.6166 - loss: 1.0007 - val_accuracy: 0.6319 - val_loss: 0.9409
Epoch 2/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 6s 188ms/step - accuracy: 0.6481 - loss: 0.9158 - val_accuracy: 0.6319 - val_loss: 0.9267
Epoch 3/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 4s 146ms/step - accuracy: 0.6481 - loss: 0.9102 - val_accuracy: 0.6319 - val_loss: 0.9029
Epoch 4/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 3s 146ms/step - accuracy: 0.6481 - loss: 0.8842 - val_accuracy: 0.6319 - val_loss: 0.9075
Epoch 5/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 6s 183ms/step - accuracy: 0.6481 - loss: 0.8809 - val_accuracy: 0.6319 - val_loss: 0.9121
Epoch 6/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 3s 142ms/step - accuracy: 0.6481 - loss: 0.8622 - val_accuracy: 0.6319 - val_loss: 0.9321
Epoch 7/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 5s 139ms/step - accuracy: 0.6981 - loss: 0.8135 - val_accuracy: 0.6258 - val_loss: 0.9487
Epoch 8/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 5s 151ms/step - accuracy: 0.7116 - loss: 0.7756 - val_accuracy: 0.6074 - val_loss: 0.9440
Epoch 9/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 3s 142ms/step - accuracy: 0.7173 - loss: 0.7177 - val_accuracy: 0.5828 - val_loss: 1.0307
Epoch 10/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 6s 178ms/step - accuracy: 0.7390 - loss: 0.6698 - val_accuracy: 0.6012 - val_loss: 1.0240
<keras.src.callbacks.history.History at 0x7ca10157e050>
```

Training Results of the Sentiment Classification Model (LSTM)

LSTM - Emotion Results

Training Accuracy:

74%

Validation Accuracy:

60%

High overfitting observed

Oorrectly predicts only Anger

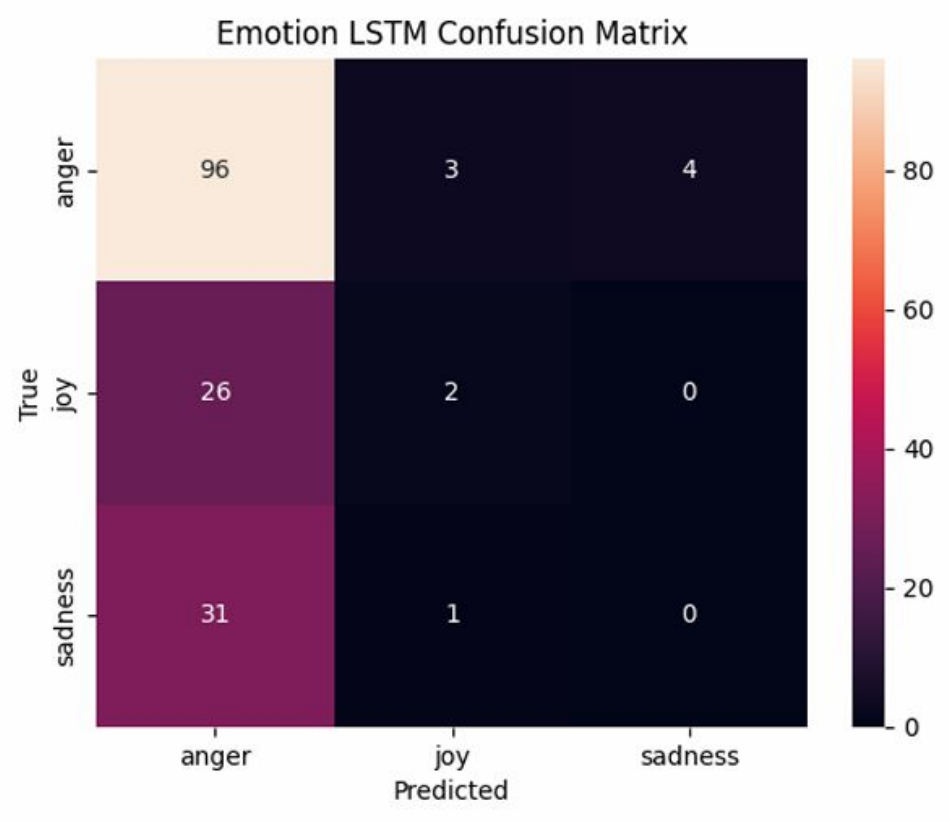
F1-Score:

Anger: 0.75

Joy: 0.12

Sadness: 0.00

LSTM - Emotion Results



LSTM - Emotion Results

Classification Report - Sentiment LSTM

	precision	recall	f1-score	support
negative	0.83	0.99	0.90	135
positive	0.00	0.00	0.00	28
accuracy			0.82	163
macro avg	0.41	0.50	0.45	163
weighted avg	0.69	0.82	0.75	163

classification Report for Emotion using (LSTM)

Discussion & Analysis

Key Insight: Both models biased toward dominant classes

Class Imbalance Impact:

-ANN handled imbalance better than LSTM

-LSTM overfitted, poor generalization on minority classes

ANN outperformed LSTM in:

-Overall Accuracy

-Balanced prediction (to some extent)

Conclusion & Future Work

Achievements:

-Built full NLP pipeline
-Developed and compared ANN vs. LSTM

Limitations:

-Dataset imbalance
-Overfitting (especially in LSTM)
-Poor recognition of "joy" and "sadness"

Future Work:

-Collect balanced dataset

-Apply SMOTE/class weights

-Try BERT or transformer models

-Add interpretability (e.g., LIME, SHAP)

-Extend analysis to other movies

Appendix

The following is selected parts of the code and project visualizations

```
#  Import Libraries
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Data preprocessing

```
[ ]  # Preprocess Text

     # Function to clean review text
     def clean_text(text):
         text = str(text).lower()                    # Convert text to lowercase
         text = re.sub(r'http\S+', '', text)         # Remove URLs
         text = re.sub(r'[^a-z\s]', '', text)        # Remove all non-alphabetic characters
         text = re.sub(r'\s+', ' ', text).strip()    # Remove extra whitespace
         return text

     # Apply the cleaning function to the 'review' column
     df['cleaned'] = df['review'].apply(clean_text)

     # Display the first 10 cleaned reviews
     df.head(10)
```

	review	Sentiment	Label (emotion)	cleaned
0	I have to say, the most sensible reviews are t...	negative	Sadness	i have to say the most sensible reviews are th...
1	The movie was OK. Definitely better than the H...	Negative	Sadness	the movie was ok definitely better than the ha...
2	I have no complaints about the acting, product...	negative	Anger	i have no complaints about the acting producti...
3	I really can't stand loose ends. So many quest...	Negative	Anger	i really cant stand loose ends so many questio...
4	There are some fantastic short films out there...	negative	Anger	there are some fantastic short films out there...
5	Why. Why. Why. Why. Just why? Why do ...	Negative	Anger	why why why why just whybrbrwhy do people find...
6	If you have seen The Happening, you have LITER...	negative	Anger	if you have seen the happening you have litera...
7	I didn't read the book before I watched the mo...	positive	Joy	i didnt read the book before i watched the mov...
8	I enjoyed this movie. It kept me from looking ...	positive	Joy	i enjoyed this movie it kept me from looking a...
9	The most common complaint I'm reading from use...	positive	Joy	the most common complaint im reading from user...

Text Representaion

```
#Text Representation using Tokenizer

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
tokenizer.fit_on_texts(df['clean_text'])

sequences = tokenizer.texts_to_sequences(df['clean_text'])
padded_sequences = pad_sequences(sequences, padding='post', maxlen=100)
```

Modeling

```
# TF-IDF Vectorization
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = vectorizer.fit_transform(df['cleaned']).toarray()

# Split data with filtered DataFrame and corresponding TF-IDF features
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(
    X_tfidf, df['sentiment_encoded'], test_size=0.2, stratify=df['sentiment_encoded'], random_state=42)

# Print shapes of the resulting datasets
print("Shape of TF-IDF matrix:", X_tfidf.shape)
print("Training set shape:", X_train_s.shape)
print("Test set shape:", X_test_s.shape)
```

```
Shape of TF-IDF matrix: (811, 5000)
Training set shape: (648, 5000)
Test set shape: (163, 5000)
```

```python
# ANN Model - Sentiment Classification
import numpy as np
import random
import tensorflow as tf

seed = 42
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)

# 2. Define and compile the ANN model
from tensorflow.keras import layers

ann_sent = tf.keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train_s.shape[1],)),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(le_sent.classes_), activation='softmax')  # Output layer
])


ann_sent.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# 3. Train the model
```

```
[ ]  # ANN Model - Emotion Classification

     import numpy as np
     import random
     import tensorflow as tf

     seed = 42
     np.random.seed(seed)
     random.seed(seed)
     tf.random.set_seed(seed)

     # Split the data (already includes fixed random_state)
     from sklearn.model_selection import train_test_split

     X_train_e, X_test_e, y_train_e, y_test_e = train_test_split(
         X_tfidf, df['emotion_encoded'], test_size=0.2, stratify=df['emotion_encoded'], random_state=42
     )

     # Define the ANN model for emotion classification
     from tensorflow.keras import layers

     ann_emo = tf.keras.Sequential([
         layers.Dense(128, activation='relu', input_shape=(X_train_e.shape[1],)),
```

```
X_train_e, X_test_e, y_train_e, y_test_e = train_test_split(
    X_tfidf, df['emotion_encoded'], test_size=0.2, stratify=df['emotion_encoded'], random_state=42
)

# Define the ANN model for emotion classification
from tensorflow.keras import layers

ann_emo = tf.keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train_e.shape[1],)),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(le_emotion.classes_), activation='softmax')  # Output layer
])

# Compile the model
ann_emo.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
ann_emo.fit(X_train_e, y_train_e, epochs=10, validation_data=(X_test_e, y_test_e))
```

```
[ ]  # Evaluate ANN Models
     def evaluate_model(y_true, y_pred_probs, label_encoder, title):
         y_pred = np.argmax(y_pred_probs, axis=1)
         print(f'\nClassification Report - {title}')
         # Get unique classes present in y_true
         unique_classes = np.unique(y_true)
         # Filter target_names to match the unique classes
         target_names = [label_encoder.classes_[i] for i in unique_classes]
         print(classification_report(y_true, y_pred, target_names=target_names))
         cm = confusion_matrix(y_true, y_pred)
         sns.heatmap(cm, annot=True, fmt='d', xticklabels=target_names, yticklabels=target_names)
         plt.title(f'{title} Confusion Matrix')
         plt.xlabel('Predicted')
         plt.ylabel('True')
         plt.show()

     evaluate_model(y_test_s, ann_sent.predict(X_test_s), le_sent, 'Sentiment ANN')
     evaluate_model(y_test_e, ann_emo.predict(X_test_e), le_emotion, 'Emotion ANN')
```

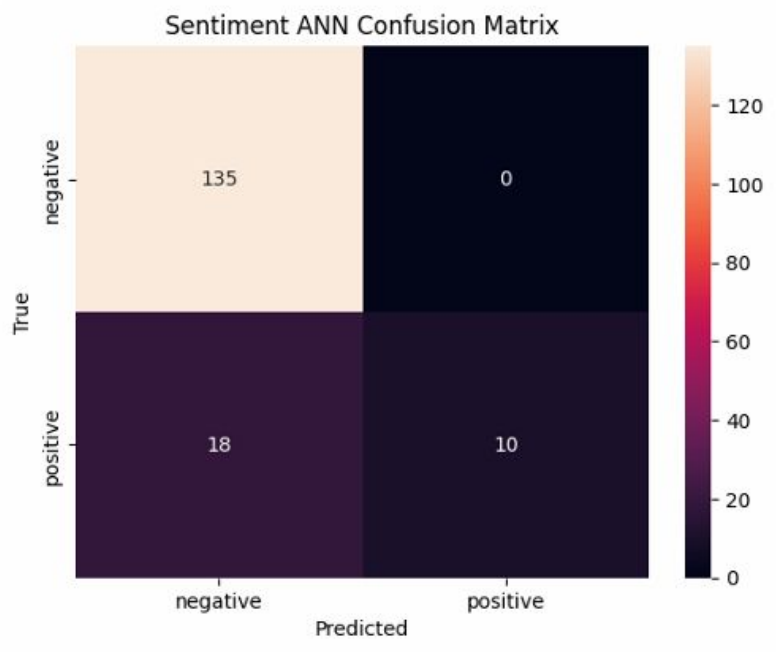
```
#  LSTM Model - Sentiment Classification


import numpy as np
import random
import tensorflow as tf

seed = 42
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)

# Imports
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer

# Filter sentiment labels that have more than 1 sample
sentiment_counts = df['sentiment_encoded'].value_counts()
df_filtered = df[df['sentiment_encoded'].isin(sentiment_counts[sentiment_counts > 1].index)].reset_index(drop=True)
```

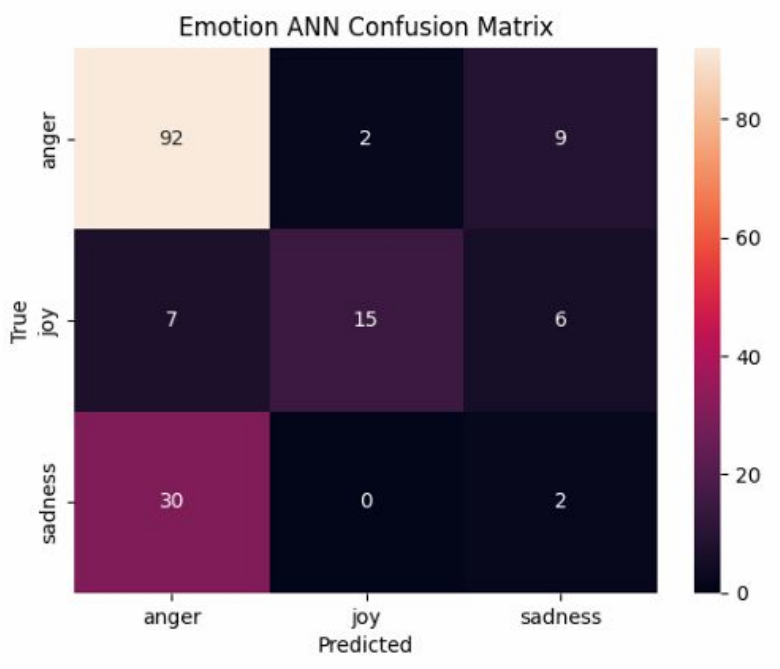
```python
# Split filtered data
X_train_s_pad, X_test_s_pad, y_train_s_pad, y_test_s_pad = train_test_split(
    padded[df_filtered.index],  # select correct padded rows
    df_filtered['sentiment_encoded'],
    test_size=0.2,
    stratify=df_filtered['sentiment_encoded'],
    random_state=42
)

# Define the LSTM model
lstm_sent = tf.keras.Sequential([
    layers.Embedding(10000, 64, input_length=100),
    layers.LSTM(128),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(le_sent.classes_), activation='softmax')  # output
])

# Compile the model
lstm_sent.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
lstm_sent.fit(X_train_s_pad, y_train_s_pad, epochs=10, validation_data=(X_test_s_pad, y_test_s_pad))
```

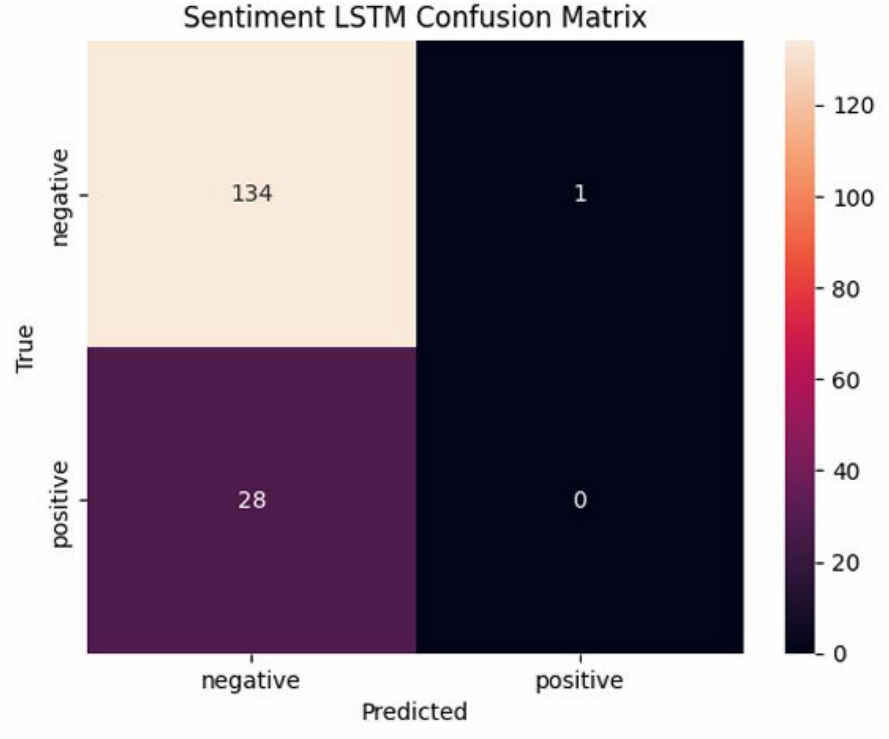
```
#  LSTM Model - Emotion Classification


import numpy as np
import random
import tensorflow as tf

seed = 42
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)

# Imports
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer

# Split the data for emotion classification
X_train_e_pad, X_test_e_pad, y_train_e_pad, y_test_e_pad = train_test_split(
    padded, df['emotion_encoded'], test_size=0.2, stratify=df['emotion_encoded'], random_state=42
)
```

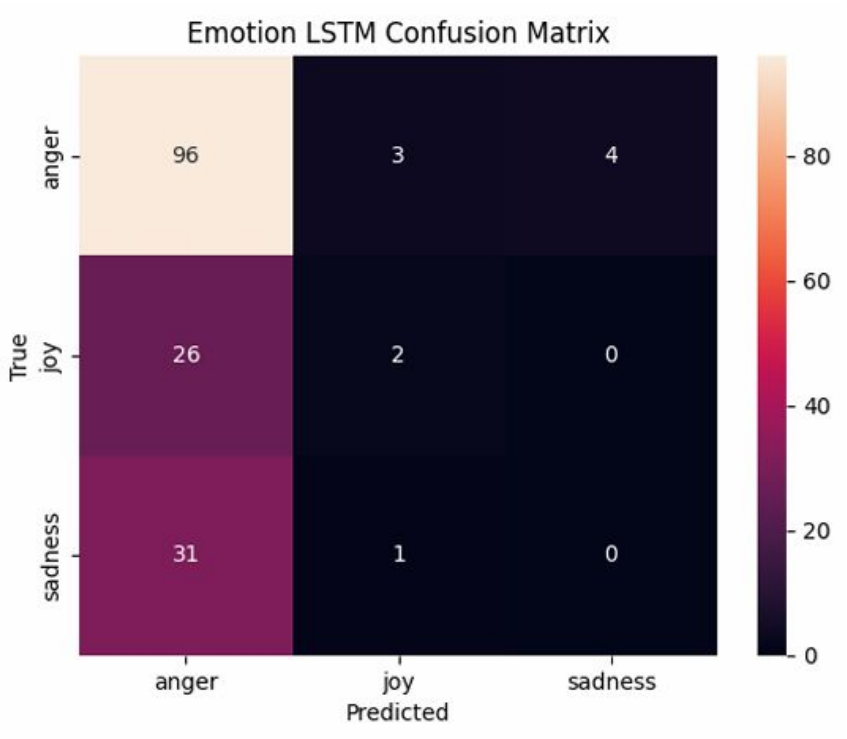
```
)

# Define the LSTM model
lstm_emo = tf.keras.Sequential([
    layers.Embedding(10000, 64, input_length=100),
    layers.LSTM(128),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(le_emotion.classes_), activation='softmax')  # Output layer
])

# Compile the model
lstm_emo.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
lstm_emo.fit(X_train_e_pad, y_train_e_pad, epochs=10, validation_data=(X_test_e_pad, y_test_e_pad))
```

```
#  Evaluate LSTM Models
evaluate_model(y_test_s_pad, lstm_sent.predict(X_test_s_pad), le_sent, 'Sentiment LSTM')
evaluate_model(y_test_e_pad, lstm_emo.predict(X_test_e_pad), le_emotion, 'Emotion LSTM')
```


Sentiment ANN Confusion Matrix
negative
positive
True
135
0
18
10
negative
positive
Predicted
120
100
80
60
40
20
0


Emotion ANN Confusion Matrix
anger
joy
sadness
True
92
2
9
7
15
6
30
0
2
anger
joy
sadness
Predicted
80
60
40
20
0


Sentiment LSTM Confusion Matrix
True
negative
positive
134
1
28
0
negative
positive
Predicted
120
100
80
60
40
20
0


Emotion LSTM Confusion Matrix
True
anger
joy
sadness
96
3
4
26
2
0
31
1
0
anger
joy
sadness
Predicted
80
60
40
20
0

Thanks

Do you have any questions?

References & Acknowledgments

IEEE DataPort IMDb Dataset

Scikit-learn, TensorFlow/Keras

Thanks to Dr. Manal Kalakatawi

Project developed in Google Colab