Basic Analysis and BERT Analysis

In [4]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter
from wordcloud import WordCloud
import time
import folium
import nbformat
import nltk
from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_

from transformers import BertTokenizer, BertForSequenceClassification, Traine
from transformers import DataCollatorWithPadding
from torch.optim import AdamW
from datasets import Dataset
import torch
from torch.utils.data import DataLoader, TensorDataset
import accelerate
from tqdm import tqdm
```

1. Read the dataset and cleaning

In [5]:
```
customer_feedback = pd.read_csv("/Users/xinyizhang/Desktop/Weill Cornell Medi
customer_feedback.head()
```

Out[5]:

	Text, Sentiment, Source, Date/Time, User ID, Location, Confidence Score
0	"I love this product!", Positive, Twitter, 202...
1	"The service was terrible.", Negative, Yelp Re...
2	"This movie is amazing!", Positive, IMDb, 2023...
3	"I'm so disappointed with their customer suppo...
4	"Just had the best meal of my life!", Positive...

In [6]:
```
# Split the columns
customer_feedback[['Text', 'Sentiment', 'Source', 'Date/Time', 'User ID', 'Lo
```

In [7]:
```
customer_feedback = customer_feedback.drop('Text, Sentiment, Source, Date/Tim
customer_feedback.head()
```

Out[7]:

	Text	Sentiment	Source	Date/Time	User ID	Location	Confidence Score
0	"I love this product!"	Positive	Twitter	2023-06-15 09:23:14	@user123	New York	0.85

	Text	Sentiment	Source	Date/Time	User ID	Location	Confidence Score
1	"The service was terrible."	Negative	Yelp Reviews	2023-06-15 11:45:32	user456	Los Angeles	0.65
2	"This movie is amazing!"	Positive	IMDb	2023-06-15 14:10:22	moviefan789	London	0.92
3	"I'm so disappointed with their customer suppo...	Negative	Online Forum	2023-06-15 17:35:11	forumuser1	Toronto	0.78
4	"Just had the best meal of my life!"	Positive	TripAdvisor	2023-06-16 08:50:59	foodie22	Paris	0.88

In [8]:
```
# In the Text column, there are many unnecessary things
customer_feedback['Text'] = customer_feedback['Text'].str.lower().str.strip()
customer_feedback['Text'] = customer_feedback['Text'].str.replace(r'[^a-zA-Z\
```

In [9]:
```
customer_feedback['Sentiment'] = customer_feedback['Sentiment'].str.strip()
customer_feedback.dropna()
customer_feedback
```

Out[9]:

	Text	Sentiment	Source	Date/Time	User ID	Location	Confidence Score
0	i love this product	Positive	Twitter	2023-06-15 09:23:14	@user123	New York	0.85
1	the service was terrible	Negative	Yelp Reviews	2023-06-15 11:45:32	user456	Los Angeles	0.65
2	this movie is amazing	Positive	IMDb	2023-06-15 14:10:22	moviefan789	London	0.92
3	im so disappointed with their customer support	Negative	Online Forum	2023-06-15 17:35:11	forumuser1	Toronto	0.78
4	just had the best meal of my life	Positive	TripAdvisor	2023-06-16 08:50:59	foodie22	Paris	0.88
...	...	...	...	...	...	...	...
93	i cant stop listening to this song its my new ...	Positive	Spotify	2023-07-03 09:17:52	musiclover789	Berlin	0.91
94	their website is so confusing and poorly designed	Negative	Website Review	2023-07-03 11:59:18	user789	Toronto	0.68
95	i had an incredible experience at	Positive	Trip Report	2023-07-03 14:40:05	thrillseeker1	Orlando	0.89

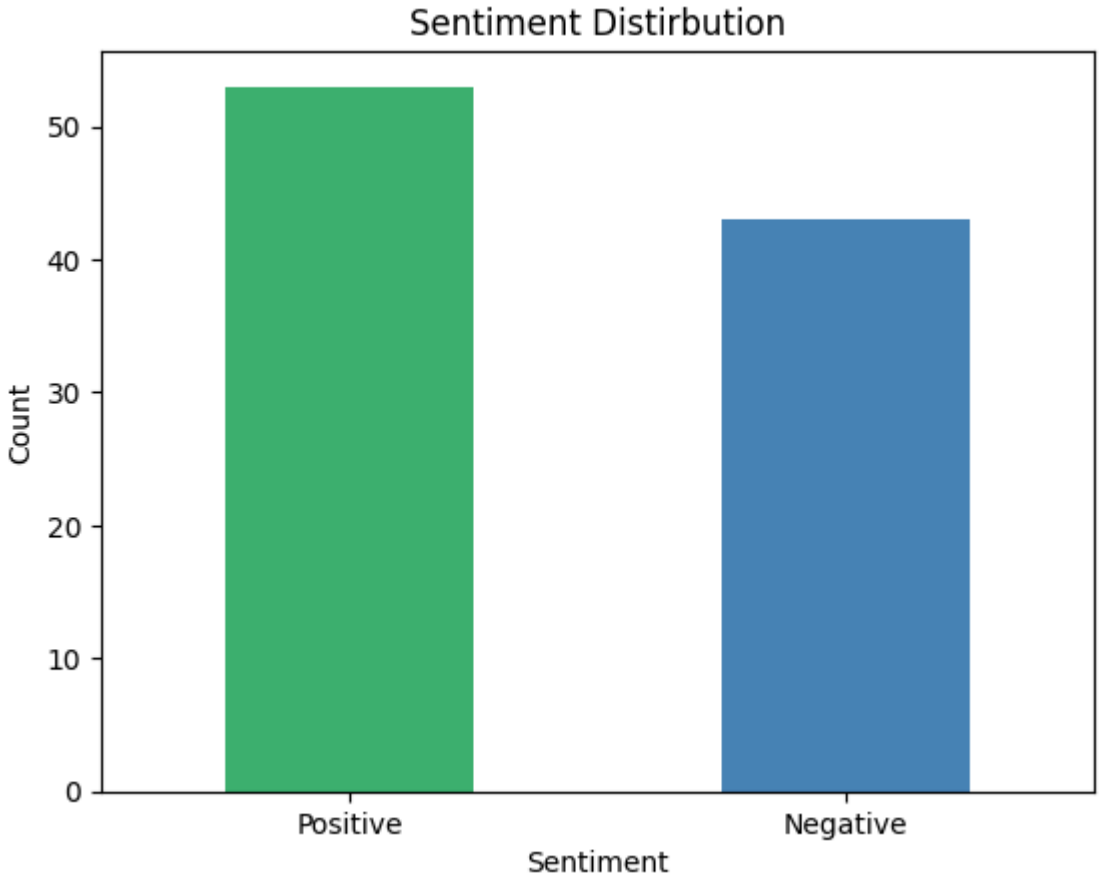
	Text	Sentiment	Source	Date/Time	User ID	Location	Confidence Score
	the theme pa...						
96	NaN	NaN	NaN	NaN	NaN	NaN	NaN
97	NaN	NaN	NaN	NaN	NaN	NaN	NaN

98 rows × 7 columns

2. Data Preprocessing

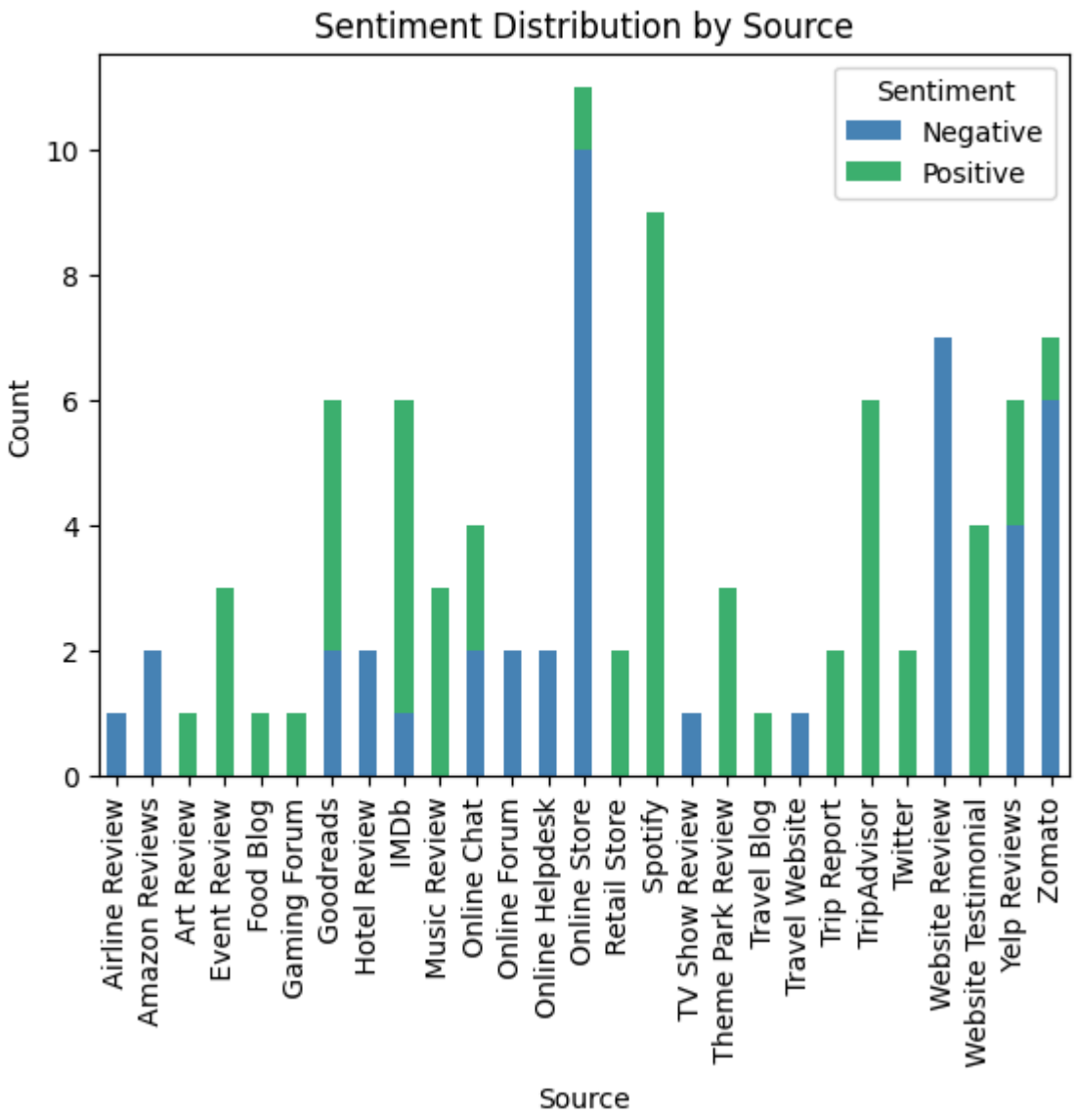
Sentiment Distribution

In [10]:
```
sentiment_counts = customer_feedback['Sentiment'].value_counts()
sentiment_counts.plot(kind='bar', color=['mediumseagreen', 'steelblue'])
plt.title("Sentiment Distirbution")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.xticks(rotation=0)
plt.show()
```



Sentiment Distribution by Source

In [11]:
```
sentiment_bygroup = customer_feedback.groupby('Source')['Sentiment'].value_co
sentiment_bygroup.plot(kind = "bar", stacked = True, color = ['steelblue', 'm
plt.title("Sentiment Distribution by Source")
plt.xlabel("Source")
plt.ylabel("Count")
plt.show()
```



Sentiment Distribution by Location

In [12]:
```
geolocator = Nominatim(user_agent="geoapi")
geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)

def get_lat_lon(city):
    try:
        loc = geolocator.geocode(city)
        return pd.Series([loc.latitude, loc.longitude]) # Get latitude and lo
    except:
        return pd.Series([None, None])

customer_feedback[['Latitude', 'Longitude']] = customer_feedback['Location'].
```

In [13]:
```
map_fig = px.scatter_geo(customer_feedback,
                         lat='Latitude',
                         lon='Longitude',
                         color='Sentiment',
                         hover_name='Location',
                         title='Overall Sentiment by City')

map_fig.update_layout(geo_scope='world')
map_fig.show()
```

In [14]:
```
sentiment_counts = (
    customer_feedback.groupby(['Location', 'Sentiment'])
    .size()
    .reset_index(name = 'Count')
)
```

In [15]:
```
locations = customer_feedback[['Location', 'Latitude', 'Longitude']].drop_dup

# Merge sentiment counts with coordinates
sentiment_counts = sentiment_counts.merge(locations, on='Location', how='left
```

In [16]:
```
fig = px.scatter_geo(sentiment_counts,
                     lat='Latitude',
                     lon='Longitude',
                     size='Count',
                     color='Sentiment',
                     hover_name='Location',
                     title='Sentiment Distribution by City (Dot Size = Count)
                     projection='natural earth')

fig.update_layout(geo_scope='world')
fig.show()
```

3. Text Analysis

In [17]:
```
# Download stopwords
nltk.download("stopwords")
stopword = set(stopwords.words('english'))

# Combine all text into one string
text = ' '.join(customer_feedback['Text'].dropna().astype(str))

# Stopwords removal
words = [word for word in text.split() if word not in stopword]
clean_text = ' '.join(words)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/xinyizhang/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Overall

In [18]:
```
wordcloud = WordCloud(background_color = 'white',
                      max_words = 200).generate(clean_text)
plt.figure(figsize = (12,6))
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

The most frequent words in this wordcloud are product, website, terrible, experience, customer, service. Probably it is more likely for people to have bad experience to comment?

In [ ]:

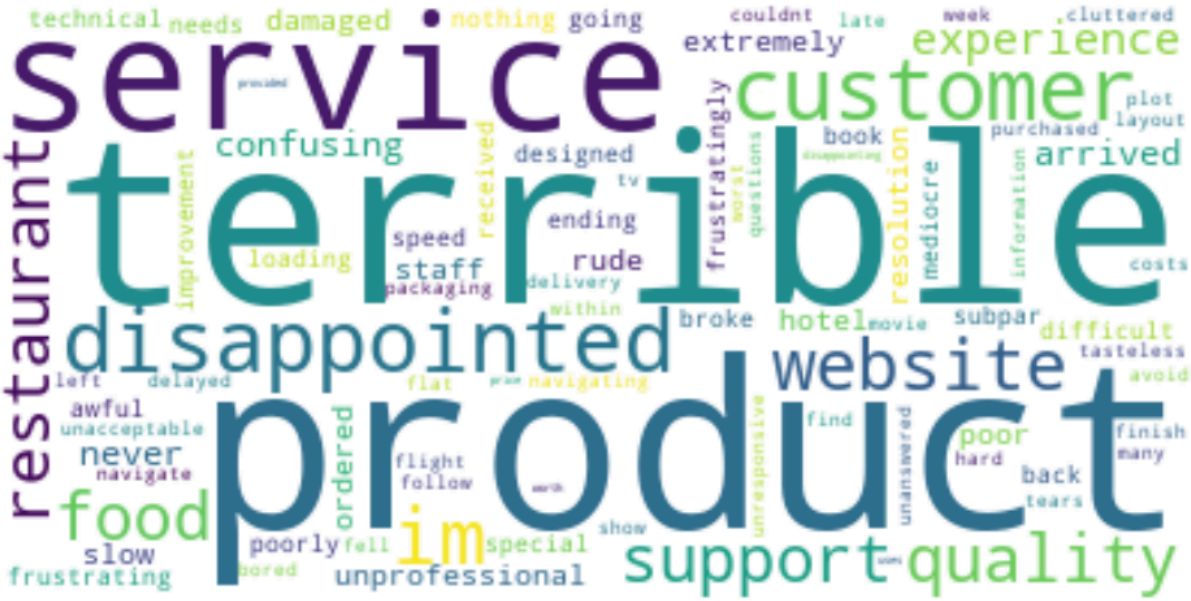
In [19]:
```
positive_word = ' '.join(customer_feedback[customer_feedback['Sentiment'] ==
negative_word = ' '.join(customer_feedback[customer_feedback['Sentiment'] ==

# Define function for positive and negative wordcloud
def senti_wordcloud(text, title, stopwords):
    wordcloud = WordCloud(background_color = 'white',
                          max_words = 200,
                          stopwords = stopwords).generate(text)
    plt.figure(figsize = (12,6))
    plt.imshow(wordcloud, interpolation = 'bilinear')
    plt.title(title, fontsize = 16)
    plt.axis('off')
    plt.show()

senti_wordcloud(positive_word, "Positive WordCloud", stopword)
senti_wordcloud(negative_word, "Negative WordCloud", stopword)
```

Positive WordCloud

Negative WordCloud



4. BERT Analysis for Sentiment Classification

Split the dataset into training set and testing set first, convert to HuggingFace Dataset

In [20]:

```
# Split data into training and testing sets
train_texts, test_texts, train_labels, test_labels = train_test_split(
    customer_feedback['Text'], customer_feedback['Sentiment'], test_size=0.2,
)

train_texts = train_texts.fillna("").astype(str)
test_texts = test_texts.fillna("").astype(str)

# Map the labels to numerical values (0 for Negative, 1 for Positive)
train_labels = train_labels.map({'Negative': 0, 'Positive': 1})
test_labels = test_labels.map({'Negative': 0, 'Positive': 1})
```

Load BERT Tokenizer and Tokenize Text

In [21]:
```
# Load pre-trained tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Tokenize the training set
train_inputs = tokenizer(
    train_texts.tolist(),  # Ensure train_texts is a list
    padding=True,
    truncation=True,
    return_tensors="pt",
    max_length=512
)

# Tokenize testing set
test_inputs = tokenizer(
    test_texts.tolist(),  # Ensure test_texts is a list
    padding=True,
    truncation=True,
    return_tensors="pt",
    max_length=512
)

# Add dynamic padding
data_collator = DataCollatorWithPadding(tokenizer = tokenizer)
```

In [22]:
```
# Convert labels to tensors
train_labels_tensor = torch.tensor(train_labels.values, dtype = torch.long) #
test_labels_tensor = torch.tensor(test_labels.values, dtype = torch.long)
```

In [23]:
```
# Train dataset and dataloader
train_dataset = TensorDataset(train_inputs['input_ids'], train_inputs['attent
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)

# Test dataset and dataloader
test_dataset = TensorDataset(test_inputs['input_ids'], test_inputs['attention
test_loader = DataLoader(test_dataset, batch_size=16)
```

Load pre-trained BERT model for classification

In [24]:
```
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", nu
optimizer = AdamW(model.parameters(), lr=2e-5)
```

```
Some weights of BertForSequenceClassification were not initialized from the m
odel checkpoint at bert-base-uncased and are newly initialized: ['classifier.
bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use
it for predictions and inference.
```

Train the model

In [25]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

epochs = 6
model.train()

for epoch in range(epochs):
    total_loss = 0
```

```
    loop = tqdm(train_loader, leave=True)
    for batch in loop:
        input_ids, attention_mask, labels = [x.to(device) for x in batch]

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask=attention_mask, labels=labe
        loss = outputs.loss
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        loop.set_description(f"Epoch {epoch+1}")
        loop.set_postfix(loss=loss.item())
    avg_loss = total_loss / len(train_loader)
    print(f"Epoch {epoch + 1}/{epochs}, Training Loss: {avg_loss}")
```

```
  0%|          | 0/5 [00:00<?, ?it/s]/Users/xinyizhang/Library/Python/3.9/li
b/python/site-packages/torch/nn/modules/module.py:1762: FutureWarning:

`encoder_attention_mask` is deprecated and will be removed in version 4.55.0
for `BertSdpaSelfAttention.forward`.

Epoch 1: 100%|██████████| 5/5 [00:05<00:00,  1.06s/it, loss=0.669]
Epoch 1/6, Training Loss: 0.6878268122673035
Epoch 2: 100%|██████████| 5/5 [00:03<00:00,  1.26it/s, loss=0.562]
Epoch 2/6, Training Loss: 0.5388875842094422
Epoch 3: 100%|██████████| 5/5 [00:04<00:00,  1.22it/s, loss=0.434]
Epoch 3/6, Training Loss: 0.45995274782180784
Epoch 4: 100%|██████████| 5/5 [00:04<00:00,  1.03it/s, loss=0.323]
Epoch 4/6, Training Loss: 0.3559182286262512
Epoch 5: 100%|██████████| 5/5 [00:08<00:00,  1.62s/it, loss=0.255]
Epoch 5/6, Training Loss: 0.25835577845573426
Epoch 6: 100%|██████████| 5/5 [00:04<00:00,  1.22it/s, loss=0.151]
Epoch 6/6, Training Loss: 0.17779999997138977
```

Evaluation

In [26]:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

# Set the model to evaluation
model.eval()

# List for storing true labels and predicted labels
predictions = []
true_labels = []

correct_predictions = 0
total_predictions = 0

with torch.no_grad():
    for batch in test_loader:
        # Unpack the batch
        input_ids, attention_mask, labels = [x.to(device) for x in batch]

        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        preds = torch.argmax(logits, dim=1)

        predictions.extend(preds.cpu().numpy())
        true_labels.extend(labels.cpu().numpy())
```

In [27]:
```
# Compute metrics
acc = accuracy_score(true_labels, predictions)
cm = confusion_matrix(true_labels, predictions)
report = classification_report(true_labels, predictions, target_names=["Negat

print(f"Accuracy: {acc:.4f}")
print("Confusion Matrix:")
print(cm)
print("\nClassification Report:")
print(report)
```

```
Accuracy: 1.0000
Confusion Matrix:
[[ 5  0]
 [ 0 15]]

Classification Report:
              precision    recall  f1-score   support

    Negative       1.00      1.00      1.00         5
    Positive       1.00      1.00      1.00        15

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

5. Predict with New Input

In [28]:
```
def new_input(user_input):
    inputs = tokenizer(
        user_input,
        padding = True,
        truncation = True,
        return_tensors="pt",
        max_length=512
    )

    input_ids = inputs['input_ids'].to(device)
    attention_mask = inputs['attention_mask'].to(device)

    model.eval()

    # Make predictions
    with torch.no_grad():
        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        preds = torch.argmax(logits, dim=1)

        predicted_class = 'Positive' if preds == 1 else 'Negative'

        return predicted_class
```

Try the model with new input

In [29]:
```
# Use the function
inference = new_input("I love this vacuum machine. It's fantastic!")
print(f"Predicted sentiment: {inference}")
```

```
Predicted sentiment: Positive
/Users/xinyizhang/Library/Python/3.9/lib/python/site-packages/torch/nn/module
s/module.py:1762: FutureWarning:

`encoder_attention_mask` is deprecated and will be removed in version 4.55.0
for `BertSdpaSelfAttention.forward`.
```

In [30]:
```
negative_comment = new_input("This product sucks. I hate it. It does not perf
print(f"Predicted sentiment: {negative_comment}")
```

```
Predicted sentiment: Negative
```

In [ ]:

LSTM Analysis

In [31]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_

import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
```

1. Prepare

In [32]:
```
# Ensure all text entries are strings
customer_feedback['Text'] = customer_feedback['Text'].astype(str)

# Encode sentiment labels (Positive = 1, Negative = 0)
le = LabelEncoder()
customer_feedback['Sentiment'] = le.fit_transform(customer_feedback['Sentimen
```

2. Split into Training and Testing

In [33]:
```
X = customer_feedback['Text']
y = customer_feedback['Sentiment']
train_texts, test_texts, train_labels, test_labels = train_test_split(X, y, t
```

3. Tokenization and Padding

In [34]:
```
max_words = 10000  # Max number of words in vocab
max_len = 512      # Max length of sequence

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(train_texts)
```

```
train_sequences = tokenizer.texts_to_sequences(train_texts)
test_sequences = tokenizer.texts_to_sequences(test_texts)

train_padded = pad_sequences(train_sequences, maxlen=max_len)
test_padded = pad_sequences(test_sequences, maxlen=max_len)
```

4. Define LSTM Model

In [35]:

```
embedding_dim = 100
lstm_units = 128  # Reduced for practicality

model = Sequential()
model.add(Embedding(input_dim=max_words, output_dim=embedding_dim, input_leng
model.add(LSTM(units=lstm_units, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accurac
```

```
/Users/xinyizhang/Library/Python/3.9/lib/python/site-packages/keras/src/layer
s/core/embedding.py:97: UserWarning:

Argument `input_length` is deprecated. Just remove it.
```

5. Train the Model

In [36]:

```
model.fit(train_padded, train_labels, epochs=6, batch_size=64, validation_spl
```

```
Epoch 1/6
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step - accuracy: 0.5936 - loss: 0.6939 - val_a
ccuracy: 0.6250 - val_loss: 0.6890
Epoch 2/6
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 721ms/step - accuracy: 0.5936 - loss: 0.6850 - va
l_accuracy: 0.6250 - val_loss: 0.6811
Epoch 3/6
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 726ms/step - accuracy: 0.4601 - loss: 0.6775 - va
l_accuracy: 0.6250 - val_loss: 0.6721
Epoch 4/6
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 725ms/step - accuracy: 0.4810 - loss: 0.6688 - va
l_accuracy: 0.6250 - val_loss: 0.6603
Epoch 5/6
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step - accuracy: 0.4757 - loss: 0.6605 - va
l_accuracy: 0.6250 - val_loss: 0.6469
Epoch 6/6
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 714ms/step - accuracy: 0.4653 - loss: 0.6428 - va
l_accuracy: 0.6250 - val_loss: 0.6293
```

Out[36]:

```
<keras.src.callbacks.history.History at 0x33b8979d0>
```

6. Evaluation

In [37]:

```
# Predict on test data
pred_probs = model.predict(test_padded)
predictions = (pred_probs > 0.5).astype("int32").flatten()

# Accuracy
acc_lstm = accuracy_score(test_labels, predictions)
print(f"Accuracy: {acc_lstm:.4f}")

# Confusion matrix
```

```
cm_lstm = confusion_matrix(test_labels, predictions)
print("Confusion Matrix:")
print(cm_lstm)

# Classification report
report_lstm = classification_report(test_labels, predictions, target_names=["
print("\nClassification Report:")
print(report_lstm)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 286ms/step
Accuracy: 0.7500
Confusion Matrix:
[[ 0  5]
 [ 0 15]]

Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         5
    Positive       0.75      1.00      0.86        15

    accuracy                           0.75        20
   macro avg       0.38      0.50      0.43        20
weighted avg       0.56      0.75      0.64        20

/Users/xinyizhang/Library/Python/3.9/lib/python/site-packages/sklearn/metric
s/_classification.py:1531: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.

/Users/xinyizhang/Library/Python/3.9/lib/python/site-packages/sklearn/metric
s/_classification.py:1531: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.

/Users/xinyizhang/Library/Python/3.9/lib/python/site-packages/sklearn/metric
s/_classification.py:1531: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
```

7. Predict on New Input

In [38]:

```
def new_input_lstm(user_input):
    seq = tokenizer.texts_to_sequences([user_input])
    padded = pad_sequences(seq, maxlen=max_len)
    prob = model.predict(padded)[0][0]
    label = "Positive" if prob > 0.5 else "Negative"
    return label

# Example
comment = new_input_lstm("I love this vacuum machine. It's fantastic!")
print(f"Predicted sentiment: {comment}")

comment = new_input_lstm("I think this tool is hard to use.")
print(f"Predicted sentiment: {comment}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 267ms/step
Predicted sentiment: Positive
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
Predicted sentiment: Positive
```

In [40]:

```
bert_metrics = {
    "accuracy": acc,
    "confusion_matrix": cm,
    "classification_report": report
}

lstm_metrics = {
    "accuracy": acc_lstm,
    "confusion_matrix": cm_lstm,
    "classification_report": report_lstm
}
```

In [41]:

```
def print_comparison(bert_metrics, lstm_metrics):
    print(f"{'Metric':<20} | {'BERT':<15} | {'LSTM':<15}")
    print("-" * 55)

    print(f"{'Accuracy':<20} | {bert_metrics['accuracy']:<15.4f} | {lstm_metr

    print("\nConfusion Matrix (BERT):")
    print(bert_metrics['confusion_matrix'])
    print("Confusion Matrix (LSTM):")
    print(lstm_metrics['confusion_matrix'])

    print("\nClassification Report (BERT):")
    print(bert_metrics['classification_report'])
    print("Classification Report (LSTM):")
    print(lstm_metrics['classification_report'])

print_comparison(bert_metrics, lstm_metrics)
```

```
Metric               | BERT            | LSTM
-------------------------------------------------------
Accuracy             | 1.0000          | 0.7500

Confusion Matrix (BERT):
[[ 5  0]
 [ 0 15]]
Confusion Matrix (LSTM):
[[ 0  5]
 [ 0 15]]

Classification Report (BERT):
              precision    recall  f1-score   support

    Negative       1.00      1.00      1.00         5
    Positive       1.00      1.00      1.00        15

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Classification Report (LSTM):
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         5
    Positive       0.75      1.00      0.86        15

    accuracy                           0.75        20
   macro avg       0.38      0.50      0.43        20
```

```
weighted avg       0.56      0.75      0.64         20
```

It seems BERT has a better performance than LSTM.