```
import pandas as pd



df = pd.read_csv('/content/twitter_training.csv')


import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
import re


df.columns = ['ID', 'Entity', 'Sentiment', 'Tweet content']
df = df.dropna(subset=['Tweet content', 'Sentiment'])
df['Tweet content'] = df['Tweet content'].apply(lambda x: re.sub(r'http\S+', '', x)) # Remove URLs
df['Tweet content'] = df['Tweet content'].apply(lambda x: re.sub(r'[^A-Za-z0-9 ]+', '', x)) # Remove special characters

sentiment_encoder = LabelEncoder()
df['Sentiment_encoded'] = sentiment_encoder.fit_transform(df['Sentiment'])

# Tokenization and Padding
tokenizer = Tokenizer(num_words=10000, oov_token='<oov>')
tokenizer.fit_on_texts(df['Tweet content'])
sequences = tokenizer.texts_to_sequences(df['Tweet content'])
padded_sequences = pad_sequences(sequences, maxlen=128, padding='post', truncating='post')

# One-Hot Encoding for Sentiments
sentiment_one_hot = to_categorical(df['Sentiment_encoded'])

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, sentiment_one_hot, test_size=0.2, random_state=42)
```

```
/tmp/ipython-input-8-757417193.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Tweet content'] = df['Tweet content'].apply(lambda x: re.sub(r'http\S+', '', x)) # Remove URLs
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

model = Sequential([
    Embedding(10000, 16, input_length=128),
    LSTM(32),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(len(sentiment_encoder.classes_), activation='softmax') # Output layer with number of sentiment classes
])

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

model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. J
  warnings.warn(
```

Model: "sequential"

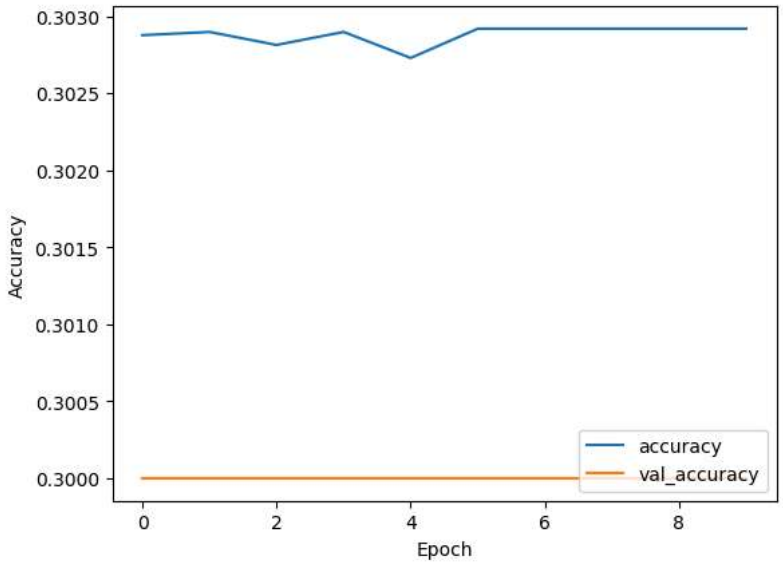
Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
lstm (LSTM)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)
dropout (Dropout)	?	0
dense_1 (Dense)	?	0 (unbuilt)

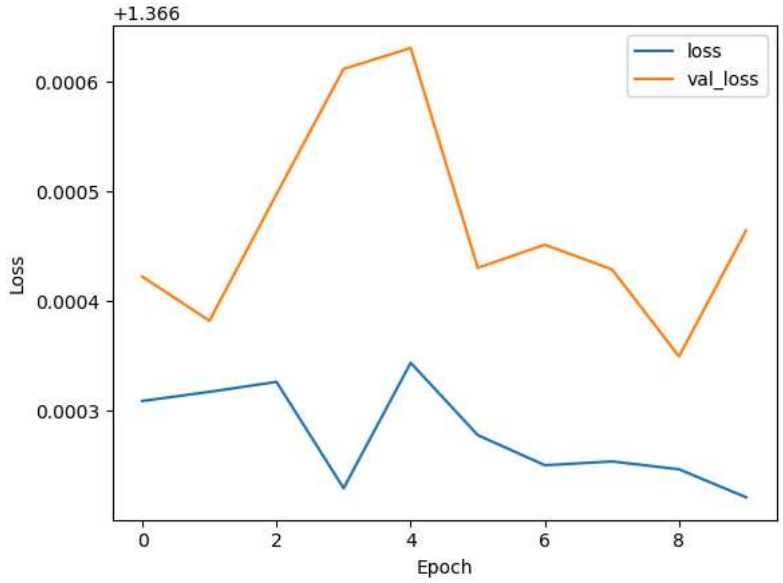
Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)

```
 import numpy as np

history = model.fit(X_train, y_train, epochs=10, validation_split=0.2)

loss, accuracy = model.evaluate(X_test, y_test)
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}')
print(f'Test Accuracy: {accuracy}')

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()

plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()

def predict_sentiment(text):
  cleaned_text = re.sub(r'http\S+', '', text)
  cleaned_text = re.sub(r'[^A-Za-z0-9 ]+', '', cleaned_text)
  sequence = tokenizer.texts_to_sequences([cleaned_text])
  padded_sequence = pad_sequences(sequence, maxlen=128, padding='post', truncating='post')
  prediction = model.predict(padded_sequence)
  predicted_class_index = np.argmax(prediction)
  predicted_sentiment = sentiment_encoder.inverse_transform([predicted_class_index])[0]
  return predicted_sentiment
```

```
Epoch 1/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 90s 61ms/step - accuracy: 0.3003 - loss: 1.3675 - val_accuracy: 0.3000 - val_loss: 1.3664
Epoch 2/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 139s 59ms/step - accuracy: 0.3041 - loss: 1.3655 - val_accuracy: 0.3000 - val_loss: 1.3664
Epoch 3/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 81s 55ms/step - accuracy: 0.3013 - loss: 1.3654 - val_accuracy: 0.3000 - val_loss: 1.3665
Epoch 4/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 85s 57ms/step - accuracy: 0.3018 - loss: 1.3671 - val_accuracy: 0.3000 - val_loss: 1.3666
Epoch 5/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 141s 57ms/step - accuracy: 0.3009 - loss: 1.3660 - val_accuracy: 0.3000 - val_loss: 1.3666
Epoch 6/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 143s 58ms/step - accuracy: 0.3031 - loss: 1.3661 - val_accuracy: 0.3000 - val_loss: 1.3664
Epoch 7/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 137s 54ms/step - accuracy: 0.3019 - loss: 1.3660 - val_accuracy: 0.3000 - val_loss: 1.3665
Epoch 8/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 81s 54ms/step - accuracy: 0.3046 - loss: 1.3650 - val_accuracy: 0.3000 - val_loss: 1.3664
Epoch 9/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 87s 58ms/step - accuracy: 0.3017 - loss: 1.3667 - val_accuracy: 0.3000 - val_loss: 1.3663
Epoch 10/10
1480/1480 ━━━━━━━━━━━━━━━━━━━━ 138s 55ms/step - accuracy: 0.3039 - loss: 1.3640 - val_accuracy: 0.3000 - val_loss: 1.3665
463/463 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - accuracy: 0.2950 - loss: 1.3695
Test Loss: 1.367875576019287
Test Accuracy: 0.3015744388103485
```






```
import matplotlib.pyplot as plt

sentiment_counts = df['Sentiment'].value_counts()
plt.figure(figsize=(8, 6))
sentiment_counts.plot(kind='bar')
plt.title('Distribution of Sentiments')
plt.xlabel('Sentiment')
plt.ylabel('Number of Tweets')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

entity_counts = df['Entity'].value_counts().head(15)
```

```
plt.figure(figsize=(12, 6))
entity_counts.plot(kind='bar')
plt.title('Distribution of Top 15 Entities')
plt.xlabel('Entity')
plt.ylabel('Number of Tweets')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

top_entities = df['Entity'].value_counts().index[:5]
for entity in top_entities:
  entity_df = df[df['Entity'] == entity]
  sentiment_by_entity = entity_df['Sentiment'].value_counts()
  plt.figure(figsize=(6, 4))
  sentiment_by_entity.plot(kind='bar')
  plt.title(f'Sentiment Distribution for {entity}')
  plt.xlabel('Sentiment')
  plt.ylabel('Number of Tweets')
  plt.xticks(rotation=45)
  plt.tight_layout()
  plt.show()
```