A logo of a university

AI-generated content may be incorrect.

COMSATS University Islamabad, Lahore campus

Submitted by:

Mubbara Majid (SP23-BAI-027)

Noor Fatima (SP23-BAI-046)

Submitted to:

Sir Zeeshan Gillani

EMOTION DETECTION Using CNN-LSTM

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

"""# Data Loading and analyzing"""

df = pd.read\_parquet("/content/train-00000-of-00001.parquet")

df.head(5)

print(f'Number of rows: {df.shape[0]}')

print(f'Number of columns: {df.shape[1]}')

df['label'].unique()

"""This is a 6-class problem."""

label\_counts = df['label'].value\_counts()

display(label\_counts)

"""# Data Preprocessing

## Check Missing and Duplicate values

"""

df.isnull().sum()

print('Duplicates: ', df.duplicated().sum())

df = df.drop\_duplicates()

print("Duplicates remaining:", df.duplicated().sum())

"""## Lowercasing"""

df['text'] = df['text'].str.lower()

df.loc[4, ['text']]

"""## Remove unnecessary elements"""

import re

def clean\_text(text):

    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # Remove URLs

    text = re.sub(r'@\w+|#\w+', '', text)    # Remove mentions and hashtags

    text = re.sub(r'[^A-Za-z\s]', '', text)  # Remove punctuations and numbers

    text = re.sub(r'\s+', ' ', text)  # Remove extra spaces

    return text.strip()

df['text'] = df['text'].apply(clean\_text)

"""## Stop words removal"""

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

stop\_words = stopwords.words('english')

df['text'] = df['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop\_words)]))

df.head(5)

"""## Check class imbalance"""

df['label'].value\_counts().plot(kind='bar')

"""The lowest class has ~10X fewer samples than the top class — this will bias the model toward the majority classes, lowering F1-score for minority emotions.

Best way is to tell the model to "care more" about underrepresented classes.

### Class Weights (Recommended for LSTM/CNN)

"""

from sklearn.utils import class\_weight

class\_weights = class\_weight.compute\_class\_weight(

    class\_weight='balanced',

    classes=np.unique(df['label']),

    y=df['label']

)

class\_weights\_dict = dict(enumerate(class\_weights))

"""The class\_weights\_dict will be used during model.fit.

## Tokenization and Padding

"""

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

MAX\_LEN = 150

VOCAB\_SIZE = 10000

tokenizer = Tokenizer(num\_words=VOCAB\_SIZE, oov\_token="<OOV>")

tokenizer.fit\_on\_texts(df['text'])

sequences = tokenizer.texts\_to\_sequences(df['text'])

padded = pad\_sequences(sequences, maxlen=MAX\_LEN, padding='post')

"""## Word2Vec Embedding Matrix"""

from google.colab import drive

drive.mount('/content/drive')

from gensim.models import KeyedVectors

model\_path = '/content/drive/MyDrive/GoogleNews-vectors-negative300.bin'

w2v\_model = KeyedVectors.load\_word2vec\_format(model\_path, binary=True)

print(w2v\_model['king'])        # Word embedding vector

print(w2v\_model.vector\_size)     # Should return 300

"""Embedding matrix"""

embedding\_dim = w2v\_model.vector\_size

word\_index = tokenizer.word\_index

embedding\_matrix = np.zeros((len(word\_index) + 1, embedding\_dim))

for word, i in word\_index.items():

    if word in w2v\_model:

        embedding\_matrix[i] = w2v\_model[word]

"""## Label Encoding"""

from keras.utils import to\_categorical

labels = to\_categorical(df['label'])

print("Padded sequences shape:", padded.shape)

print("Embedding matrix shape:", embedding\_matrix.shape)

print("Labels shape:", labels.shape)

"""# Build and Train model"""

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Bidirectional, BatchNormalization

from tensorflow.keras.optimizers import Adam

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(padded, labels, test\_size=0.2, random\_state=42, stratify=labels)

model = Sequential()

model.add(Embedding(

    input\_dim=len(word\_index) + 1,

    output\_dim=embedding\_dim,

    weights=[embedding\_matrix],

    input\_length=MAX\_LEN,

    trainable=False

))

model.add(Conv1D(filters=128, kernel\_size=5, activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(MaxPooling1D(pool\_size=2))

model.add(Conv1D(filters=64, kernel\_size=3, activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(Bidirectional(LSTM(64, return\_sequences=True)))

model.add(Dropout(0.5))

model.add(Bidirectional(LSTM(32)))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(BatchNormalization())

model.add(Dense(6, activation='softmax'))

model.compile(loss='categorical\_crossentropy',

              optimizer=Adam(learning\_rate=0.001),

              metrics=['accuracy'])

model.summary()

history = model.fit(

    X\_train, y\_train,

    validation\_data=(X\_test, y\_test),

    epochs=10,

    batch\_size=128,

    class\_weight=class\_weights\_dict,

    verbose=1

)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Val Accuracy')

plt.title("Model Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Val Loss')

plt.title("Model Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

y\_hat\_probs = model.predict(X\_test)

y\_pred = np.argmax(y\_hat\_probs, axis=1)

y\_true = np.argmax(y\_test, axis=1)

report = classification\_report(y\_true, y\_pred, digits=4)

print("Classification Report:\n", report)

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

conf\_matrix = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix")

plt.xlabel("Predicted Labels")

plt.ylabel("True Labels")

plt.show()

"""## Summary

The project builds a 6-class text classifier using Word2Vec embeddings with a CNN + Bi-LSTM model in Keras.

It achieved an overall accuracy of 93.75%, with a macro F1-score of 91.18% and weighted F1-score of 93.92%.

The model handles class imbalance well and generalizes effectively across all classes.

"""