ICP 6

Name : PRASHANTH REDDY KOPPULA
700# : 700761149

GitHub:
https://github.com/Nagi-131/700761149_ICP6

Vedio Link:
https://drive.google.com/file/d/1llfT8soAlI0Gh7c_reykZeD64UvKYvSP/view?usp=drive_link

```
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (0.37.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0) (2.15.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.9.1) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.9.1) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.9.1) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.9.1) (1.4.5)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.9.1) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.9.1) (3.1.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.5.1) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.5.1) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.5.1) (3.5.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow==2.15.0) (0.43.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow==2.15.0) (1.2.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.6)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow==2.15.0) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.0.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (5.4.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2024.7.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.1.5)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.
```

```python
import pandas as pd # Basic packages for creating dataframes and loading dataset
import numpy as np
import matplotlib.pyplot as plt # Package for visualization
import re # importing package for Regular expression operations
from sklearn.model_selection import train_test_split # Package for splitting the data
from sklearn.preprocessing import LabelEncoder # Package for conversion of categorical to Numerical
from tensorflow.keras.preprocessing.text import Tokenizer # Tokenization
from tensorflow.keras.preprocessing.sequence import pad_sequences # Add zeros or crop based on the length
from tensorflow.keras.models import Sequential # Sequential Neural Network
from tensorflow.keras.layers import Dense, Embedding, LSTM, SpatialDropout1D # For layers in Neural Network
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model

# Load the dataset as a Pandas DataFrame
path_to_csv = '/content/sample_data/Sentiment.csv'
dataset = pd.read_csv(path_to_csv, header=0)

# Select only the necessary columns 'text' and 'sentiment'
mask = dataset.columns.isin(['text', 'sentiment'])
data = dataset.loc[:, mask]

# Keeping only the necessary columns
data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-Z0-9\s]', '', x))

for idx, row in data.iterrows():
    row[0] = row[0].replace('rt', ' ') # Removing Retweets

max_features = 2000
tokenizer = Tokenizer(num_words=max_features, split=' ') # Maximum words is 2000 to tokenize sentence
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values) # Taking values to feature matrix
X = pad_sequences(X) # Padding the feature matrix

embed_dim = 128 # Dimension of the Embedded layer
lstm_out = 196 # Long short-term memory (LSTM) layer neurons
```

```
def createmodel():
    model = Sequential() # Sequential Neural Network
    model.add(Embedding(max_features, embed_dim, input_length = X.shape[1])) # input dimension 2000 Neurons, output dimension 128 Neurons
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2)) # Drop out 20%, 196 output Neurons, recurrent dropout 20%
    model.add(Dense(3, activation='softmax')) # 3 output neurons[positive, Neutral, Negative], softmax as activation
    model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics = ['accuracy']) # Compiling the model
    return model

labelencoder = LabelEncoder() # Applying label Encoding on the label matrix
integer_encoded = labelencoder.fit_transform(data['sentiment']) # Fitting the model
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42) # 67% training data, 33% test data split

batch_size = 32 # Batch size 32
model = createmodel() # Function call to Sequential Neural Network
model.fit(X_train, Y_train, epochs=1, batch_size=batch_size, verbose=2) # verbose the higher, the more messages
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size) # evaluating the model
print(score)
print(acc)
print(model.metrics_names) # metrics of the model
print(integer_encoded)
print(data['sentiment'])

# Predicting on the text data
sentence = ['A lot of good things are happening. We are respected again throughout the world, and that is a great thing.@realDonaldTrump']
sentence = tokenizer.texts_to_sequences(sentence) # Tokenizing the sentence
sentence = pad_sequences(sentence, maxlen=X.shape[1], dtype='int32', value=0) # Padding the sentence
sentiment_probs = model.predict(sentence, batch_size=1, verbose=2)[0] # Predicting the sentence text
sentiment = np.argmax(sentiment_probs)

print(sentiment_probs)
if sentiment == 0:
    print("Neutral")
elif sentiment == 1:
    print("Negative")
else:
    print("Positive")
```

```
# Custom wrapper for Keras model
from sklearn.base import BaseEstimator, ClassifierMixin

class CustomKerasClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, build_fn=None, epochs=1, batch_size=32, verbose=1, **sk_params):
        self.build_fn = build_fn
        self.epochs = epochs
        self.batch_size = batch_size
        self.verbose = verbose
        self.sk_params = sk_params
        self.model = None

    def fit(self, X, y, **kwargs):
        self.model = self.build_fn()
        return self.model.fit(X, y, epochs=self.epochs, batch_size=self.batch_size, verbose=self.verbose, **kwargs)

    def predict(self, X, **kwargs):
        return self.model.predict(X, **kwargs)

    def predict_proba(self, X, **kwargs):
        return self.model.predict(X, **kwargs)

    def score(self, X, y, **kwargs):
        _, accuracy = self.model.evaluate(X, y, verbose=0)
        return accuracy

# Use the custom Keras classifier
model = CustomKerasClassifier(build_fn=createmodel, verbose=2)
batch_size = [10, 20, 40]
epochs = [1, 2]
param_grid = {'batch_size': batch_size, 'epochs': epochs}
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid_result = grid.fit(X_train, Y_train)

# Summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Epoch 2/2
744/744 - 91s - loss: 0.6657 - accuracy: 0.7185 - 91s/epoch - 122ms/step
372/372 - 54s - loss: 0.8349 - accuracy: 0.6426 - 54s/epoch - 145ms/step
372/372 - 55s - loss: 0.8274 - accuracy: 0.6411 - 55s/epoch - 147ms/step
372/372 - 55s - loss: 0.8271 - accuracy: 0.6464 - 55s/epoch - 147ms/step
372/372 - 51s - loss: 0.8333 - accuracy: 0.6397 - 51s/epoch - 138ms/step
372/372 - 57s - loss: 0.8265 - accuracy: 0.6418 - 57s/epoch - 153ms/step
Epoch 1/2
372/372 - 54s - loss: 0.8348 - accuracy: 0.6473 - 54s/epoch - 145ms/step
Epoch 2/2
372/372 - 53s - loss: 0.6786 - accuracy: 0.7147 - 53s/epoch - 141ms/step
Epoch 1/2
372/372 - 51s - loss: 0.8221 - accuracy: 0.6481 - 51s/epoch - 136ms/step
Epoch 2/2
372/372 - 51s - loss: 0.6802 - accuracy: 0.7132 - 51s/epoch - 136ms/step
Epoch 1/2
372/372 - 55s - loss: 0.8315 - accuracy: 0.6387 - 55s/epoch - 147ms/step
Epoch 2/2
372/372 - 52s - loss: 0.6815 - accuracy: 0.7151 - 52s/epoch - 140ms/step
Epoch 1/2
372/372 - 57s - loss: 0.8280 - accuracy: 0.6397 - 57s/epoch - 152ms/step
Epoch 2/2
372/372 - 51s - loss: 0.6709 - accuracy: 0.7130 - 51s/epoch - 138ms/step
Epoch 1/2
372/372 - 53s - loss: 0.8381 - accuracy: 0.6385 - 53s/epoch - 144ms/step
Epoch 2/2
372/372 - 49s - loss: 0.6746 - accuracy: 0.7116 - 49s/epoch - 132ms/step
186/186 - 33s - loss: 0.8493 - accuracy: 0.6321 - 33s/epoch - 177ms/step
186/186 - 35s - loss: 0.8428 - accuracy: 0.6367 - 35s/epoch - 186ms/step
186/186 - 36s - loss: 0.8437 - accuracy: 0.6357 - 36s/epoch - 192ms/step
186/186 - 33s - loss: 0.8503 - accuracy: 0.6351 - 33s/epoch - 178ms/step
186/186 - 35s - loss: 0.8431 - accuracy: 0.6366 - 35s/epoch - 187ms/step
Epoch 1/2
186/186 - 35s - loss: 0.8579 - accuracy: 0.6297 - 35s/epoch - 186ms/step
Epoch 2/2
186/186 - 31s - loss: 0.6950 - accuracy: 0.7006 - 31s/epoch - 167ms/step
Epoch 1/2
186/186 - 33s - loss: 0.8357 - accuracy: 0.6404 - 33s/epoch - 176ms/step
Epoch 2/2
186/186 - 31s - loss: 0.6883 - accuracy: 0.7062 - 31s/epoch - 168ms/step
Epoch 1/2
```

```
372/372 - 57s - loss: 0.8280 - accuracy: 0.6397 - 57s/epoch - 152ms/step
Epoch 2/2
372/372 - 51s - loss: 0.6709 - accuracy: 0.7130 - 51s/epoch - 138ms/step
Epoch 1/2
372/372 - 53s - loss: 0.8381 - accuracy: 0.6385 - 53s/epoch - 144ms/step
Epoch 2/2
372/372 - 49s - loss: 0.6746 - accuracy: 0.7116 - 49s/epoch - 132ms/step
186/186 - 33s - loss: 0.8493 - accuracy: 0.6321 - 33s/epoch - 177ms/step
186/186 - 35s - loss: 0.8428 - accuracy: 0.6367 - 35s/epoch - 186ms/step
186/186 - 36s - loss: 0.8437 - accuracy: 0.6357 - 36s/epoch - 192ms/step
186/186 - 33s - loss: 0.8503 - accuracy: 0.6351 - 33s/epoch - 178ms/step
186/186 - 35s - loss: 0.8431 - accuracy: 0.6366 - 35s/epoch - 187ms/step
Epoch 1/2
186/186 - 35s - loss: 0.8579 - accuracy: 0.6297 - 35s/epoch - 186ms/step
Epoch 2/2
186/186 - 31s - loss: 0.6950 - accuracy: 0.7006 - 31s/epoch - 167ms/step
Epoch 1/2
186/186 - 33s - loss: 0.8357 - accuracy: 0.6404 - 33s/epoch - 176ms/step
Epoch 2/2
186/186 - 31s - loss: 0.6883 - accuracy: 0.7062 - 31s/epoch - 168ms/step
Epoch 1/2
186/186 - 35s - loss: 0.8452 - accuracy: 0.6321 - 35s/epoch - 187ms/step
Epoch 2/2
186/186 - 30s - loss: 0.6822 - accuracy: 0.7117 - 30s/epoch - 162ms/step
Epoch 1/2
186/186 - 37s - loss: 0.8445 - accuracy: 0.6344 - 37s/epoch - 196ms/step
Epoch 2/2
186/186 - 30s - loss: 0.6826 - accuracy: 0.7061 - 30s/epoch - 164ms/step
Epoch 1/2
186/186 - 33s - loss: 0.8384 - accuracy: 0.6354 - 33s/epoch - 178ms/step
Epoch 2/2
186/186 - 31s - loss: 0.6760 - accuracy: 0.7176 - 31s/epoch - 165ms/step
Epoch 1/2
233/233 - 45s - loss: 0.8322 - accuracy: 0.6394 - 45s/epoch - 194ms/step
Epoch 2/2
233/233 - 39s - loss: 0.6823 - accuracy: 0.7097 - 39s/epoch - 169ms/step
Best: 0.678682 using {'batch_size': 40, 'epochs': 2}
```