NLP Assignment-4

Name:P.Rahul

Roll No:2203A51439

Batch:12

- 1. Load data from keras.datasets and perform following computational analysis:- [CO2]
- (a) Preprocessing of the Data
- (b) Divide data into training and testing data set
- (c) Build the Recurrent Neural network (RNN) Model
- (d) Training the RNN Model
- (e) Evaluate the model on the test dataset to see how well it generalizes.
- 2. Develop LSTM (Long Short-Term Memory) by utilizing data set from https://www.kaggle.com/code/amirrezaeian/time-series-data-analysis-using-lstm-tutorial Links to an external site. or take any time series data. [CO2]
- 3. Demonstrate Vanishing and Exploding Gradients on deep neural network. [CO2]

Task-1:

(a) Preprocessing of the Data

```
Task 1: RNN with IMDB Dataset

from keras.datasets import imdb
from keras.preprocessing import sequence

# Load the dataset
max_words = 10000 # Top 10,000 most frequent words in the dataset
maxlen = 500 # Limit the review length to 500 words

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_words)

# Preprocessing: Pad sequences to ensure uniform length
x_train = sequence.pad_sequence
x_test = sequence.pad_sequence
print(f'Training data shape: {x_train.shape}')

print(f'Testing data shape: {x_test.shape}')

Training data shape: (25000, 500)
Testing data shape: (25000, 500)
```

(b) Divide data into training and testing data set

The data is already divided into training and testing data set

(c) Build the Recurrent Neural network (RNN) Model

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Dense, Embedding

# RNN Model
model = Sequential()
model.add(Embedding(max words, 128)) # Embedding layer to convert words into vectors
model.add(SimpleRNN(64)) # Simple RNN with 64 units
model.add(Dense(1, activation='sigmoid')) # Output layer with a single neuron for binary classification
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model: "sequential_1"

Layer (type) Output Shape Param #
embedding_1 (Embedding) ? @ (unbuilt)

simple_rnn_1 (simpleRNN) ? @ (unbuilt)

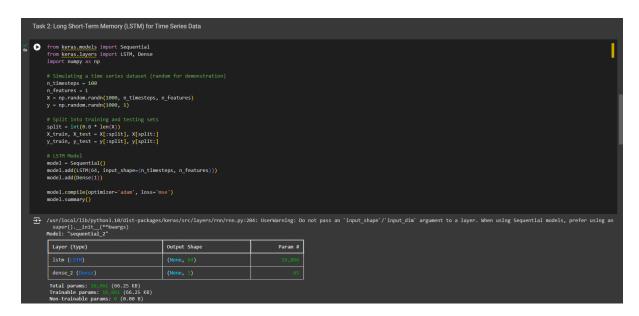
dense_1 (Dense) ? @ (unbuilt)

Total params: @ (0.00 B)
Trainable params: @ (0.00 B)
Trainable params: @ (0.00 B)
Non-trainable params: @ (0.00 B)
Non-trainable params: @ (0.00 B)
```

(d) Training the RNN Model

(e) Evaluate the model on the test dataset to see how well it generalizes.

Task-2:



```
[13] # Evaluate the model
test_loss = model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss: .4f}')

The structure of the following prints of the structure of the struct
```

Task-3:

```
Task 3: Demonstrate Vanishing and Exploding Gradients in Deep Neural Networks

from keras.models import Sequential from keras.alayers import Dense from keras.optimizers import Dense from keras.optimizers import SGO import tensorflow as tf

# Build a deep neural network with many layers model = Sequential() model.add(Dense(64, activation='tanh', input_shape=(n_timesteps,)))

# Add many layers to show the vanishing/exploding gradient problem for _ in range(50): # 50 layers model.add(Dense(64, activation='tanh'))

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

# Using Stochastic Gradient Descent (SGO) to highlight the gradient problem optimizer = SGO(learning.rate=0.01) model.compile(optimizer-optimizer, loss='mse')

**Just/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a lasuper().__init_(activity_regularizer=activity_regularizer, **kwargs)
```

```
c (2) Import tensorfiow as if

# Conton Caliback to Nontion Gradients
class Gradienthon(reff keras.caliback);
dr or addreshino(reff keras.caliback);
dr or addreshino(ref
```