CHANDIGARH UNIVERSITY
Discover. Learn. Empower.
NAAC GRADE A+ Accredited University

Deep Learning

Subject Code:22CAP-771

Student Name: Shubham Kumar Dutta **UID:22MCI10172**

Section/Group:22MAM-1_B **Semester:4th**

Experiment No. 9

1.) Take NLP set from Kaggle, preprocess data set (cleanse, tokenize, create vocabulary, convert to sequence, pad to fixed length) and Build RNN and LSTM models to demonstrate prediction with keras.

Implementation :-

In [1]:
```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, SimpleRNN, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Sample data
sentences = ['I love coding', 'Machine learning is fascinating']

# Tokenizing the sentences
tokenizer = Tokenizer()
tokenizer.fit_on_texts(sentences)
sequences = tokenizer.texts_to_sequences(sentences)
print(sequences)
```

```
[[1, 2, 3], [4, 5, 6, 7]]
```

In [2]:
```
X = []
y = []
for sequence in sequences:
    for i in range(1, len(sequence)):
        X.append(sequence[:i])
        y.append(sequence[i])
```

In [3]:
```
max_len = max([len(seq) for seq in X])
X = pad_sequences(X, maxlen=max_len, padding='pre')
y = np.array(y)
```

In [4]:
```
rnn_model = Sequential([
    Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=50,),
    SimpleRNN(64),
    Dense(len(tokenizer.word_index)+1, activation='softmax')
])
```

In [5]:
```
rnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
rnn_model.fit(X, y, epochs=10, batch_size=32)

```

```
Epoch 1/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - accuracy: 0.0000e+00 - loss: 2.0580
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.2000 - loss: 2.0212
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.4000 - loss: 1.9845
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.8000 - loss: 1.9477
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.8000 - loss: 1.9106
```

```
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 0.8000 - loss: 1.8729
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.8000 - loss: 1.8347
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.8000 - loss: 1.7956
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.8000 - loss: 1.7557
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.8000 - loss: 1.7147
```

Out[5]: <keras.src.callbacks.history.History at 0x1cf6bf272e0>

In [6]:
```
# LSTM Model
lstm_model = Sequential([
    Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=50,),
    LSTM(64),
    Dense(len(tokenizer.word_index)+1, activation='softmax')
])

```

In [7]:
```
lstm_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
lstm_model.fit(X, y, epochs=10, batch_size=32)
```

```
Epoch 1/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 0.2000 - loss: 2.0813
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step - accuracy: 0.0000e+00 - loss: 2.0752
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.2000 - loss: 2.0691
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.2000 - loss: 2.0630
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - accuracy: 0.2000 - loss: 2.0568
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - accuracy: 0.2000 - loss: 2.0505
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.2000 - loss: 2.0440
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.6000 - loss: 2.0374
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.6000 - loss: 2.0305
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.6000 - loss: 2.0234
```

Out[7]: <keras.src.callbacks.history.History at 0x1cf6d9123b0>

Learning outcomes (What I have learnt):

1. Learned how to import Keras and LSTM.
2. Learned how to Tokenize the sentence.
3. Learned how to build RNN and LSTM models.
4. Understood the concept of sparse_categorical_crossentropy.
5. Understood working of LSTM and RNN.