NAME: Divyesh Khunt **Sapid:60009210116** **Batch:D12**

Experiment no 3:

Lab exercise to be performed:

Consider fake news data set

Data Preprocessing

1. Load the Dataset
2. Text Cleaning:

   Clean the text data (remove special characters, convert to lowercase, etc.).
3. Tokenization and Padding:

   Tokenize and pad the text sequences.
4. Split Dataset:

   Split the data into training and test sets.

2.Building the RNN Model

3. Building the LSTM Model

4. Building GRU Model

5. Perform Fake News Classification for English News

]:
```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/fake-news-dataset/WELFake_Dataset.csv
```

]:
```
import pandas as pd
import numpy as np
import re
import string
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
```

In [3]:
```
data = pd.read_csv('/kaggle/input/fake-news-dataset/WELFake_Dataset.csv')  # Adjust the path if needed
data.head()
```

Out[3]:

	Unnamed: 0	title	text	label
0	0	LAW ENFORCEMENT ON HIGH ALERT Following Threat...	No comment is expected from Barack Obama Membe...	1
1	1	NaN	Did they post their votes for Hillary already?	1
2	2	UNBELIEVABLE! OBAMA'S ATTORNEY GENERAL SAYS MO...	Now, most of the demonstrators gathered last ...	1
3	3	Bobby Jindal, raised Hindu, uses story of Chri...	A dozen politically active pastors came here f...	0
4	4	SATAN 2: Russia unvelis an image of its terrif...	The RS-28 Sarmat missile, dubbed Satan 2, will...	1

```
data = data.dropna(subset=['text'])
data.loc[:, 'content'] = data['title'].fillna('') + ' ' + data['text']
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 72095 entries, 0 to 72133
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  72095 non-null  int64 
 1   title       71537 non-null  object
 2   text        72095 non-null  object
 3   label       72095 non-null  int64 
 4   content     72095 non-null  object
dtypes: int64(2), object(3)
```

```
import re
import string

def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(f"[{string.punctuation}]", "", text)  # Remove punctuation
    text = re.sub(r'\d+', '', text)  # Remove digits
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra spaces
    return text

data['cleaned_content'] = data['content'].apply(clean_text)
data['cleaned_content'].head()
```

```
0    law enforcement on high alert following threat...
1        did they post their votes for hillary already
2    unbelievable obama’s attorney general says mos...
3    bobby jindal raised hindu uses story of christ...
4    satan russia unvelis an image of its terrifyin...
Name: cleaned_content, dtype: object
```

```
print(np.unique(data['label'].value_counts()))
```

```
[35028 37106]
```

```
# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, data['label'], test_size=0.2, random_state=42
X_train
```

```
array([[   0,    0,    0, ...,  564,   18, 1483],
       [   0,    0,    0, ...,    6,    5,  221],
       [3712,    9, 3264, ...,   81,   21,  581],
       ...,
       [   2,    1,   73, ...,  648, 1408,  418],
       [  37,   22,    1, ...,   78,   26, 2935],
       [   1,  292,  200, ...,    9, 2119,  482]], dtype=int32)
```

```
data.head()
```

	Unnamed: 0	title	text	label	content	cleaned_content
0	0	LAW ENFORCEMENT ON HIGH ALERT Following Threat...	No comment is expected from Barack Obama Membe...	1	LAW ENFORCEMENT ON HIGH ALERT Following Threat...	law enforcement on high alert following threat...
1	1	NaN	Did they post their votes for Hillary already?	1	Did they post their votes for Hillary already?	did they post their votes for hillary already
2	2	UNBELIEVABLE! OBAMA'S ATTORNEY GENERAL SAYS MO...	Now, most of the demonstrators gathered last ...	1	UNBELIEVABLE! OBAMA'S ATTORNEY GENERAL SAYS MO...	unbelievable obama's attorney general says mos...
3	3	Bobby Jindal, raised Hindu, uses story of Chri...	A dozen politically active pastors came here f...	0	Bobby Jindal, raised Hindu, uses story of Chri...	bobby jindal raised hindu uses story of christ...
4	4	SATAN 2: Russia unvelis an image of its terrif...	The RS-28 Sarmat missile, dubbed Satan 2, will...	1	SATAN 2: Russia unvelis an image of its terrif...	satan russia unvelis an image of its terrifyin...

RNN

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(data['cleaned_content'])
sequences = tokenizer.texts_to_sequences(data['cleaned_content'])

padded_sequences = pad_sequences(sequences, maxlen=100)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, data['label'], test_size=0.2, random_state=42)
```

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

rnn_model = Sequential()
rnn_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
rnn_model.add(SimpleRNN(128, return_sequences=False))
rnn_model.add(Dense(1, activation='sigmoid'))

rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
rnn_model.summary()

rnn_model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))
```

```
Epoch 1/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 41s 42ms/step - accuracy: 0.8485 - loss: 0.3320 - val_accuracy: 0.8007 - val_loss: 0.3946
Epoch 2/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 38s 42ms/step - accuracy: 0.8787 - loss: 0.2763 - val_accuracy: 0.8152 - val_loss: 0.3732
Epoch 3/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 38s 42ms/step - accuracy: 0.8864 - loss: 0.2681 - val_accuracy: 0.9102 - val_loss: 0.2428
Epoch 4/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 38s 42ms/step - accuracy: 0.9146 - loss: 0.2138 - val_accuracy: 0.9072 - val_loss: 0.2488
Epoch 5/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 38s 42ms/step - accuracy: 0.9273 - loss: 0.1863 - val_accuracy: 0.9170 - val_loss: 0.2235
```

```
[10]: rnn_model.evaluate(X_test, y_test)
```

```
451/451 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.9162 - loss: 0.2211
```

```
ut[10]: [0.22349043190479279, 0.9169845581054688]
```

LSTM

```
# Build the LSTM model
lstm_model = Sequential()
lstm_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
lstm_model.add(LSTM(128, return_sequences=False))
lstm_model.add(Dense(1, activation='sigmoid'))

lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc
lstm_model.summary()
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/core/embedding.py:90: UserWarn
  warnings.warn(
```

Model: "sequential"

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

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

```
# Train the LSTM model
lstm_model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))
```

```
Epoch 1/5
895/895 ━━━━━━━━━━━━━━━━━━━━ 135s 148ms/step - accuracy: 0.8723 - loss: 0.2833 - val_accuracy: 0.9322 - val_loss: 0.1725
Epoch 2/5
895/895 ━━━━━━━━━━━━━━━━━━━━ 132s 147ms/step - accuracy: 0.9497 - loss: 0.1341 - val_accuracy: 0.9374 - val_loss: 0.1655
Epoch 3/5
895/895 ━━━━━━━━━━━━━━━━━━━━ 132s 147ms/step - accuracy: 0.9628 - loss: 0.1039 - val_accuracy: 0.9432 - val_loss: 0.1669
Epoch 4/5
895/895 ━━━━━━━━━━━━━━━━━━━━ 133s 148ms/step - accuracy: 0.9719 - loss: 0.0793 - val_accuracy: 0.9422 - val_loss: 0.1720
Epoch 5/5
895/895 ━━━━━━━━━━━━━━━━━━━━ 133s 149ms/step - accuracy: 0.9808 - loss: 0.0529 - val_accuracy: 0.9337 - val_loss: 0.1976
<keras.src.callbacks.history.History at 0x78db953d1cf0>
```

```
lstm_model.evaluate(X_test, y_test)
```

```
448/448 ━━━━━━━━━━━━━━━━━━━━ 18s 41ms/step - accuracy: 0.9326 - loss: 0.2039
[0.19760197401046753, 0.9336734414100647]
```

GRU

```
# Build the GRU model
gru_model = Sequential()
gru_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
gru_model.add(GRU(128, return_sequences=False))  # GRU layer with 128 units
gru_model.add(Dense(1, activation='sigmoid'))  # Output layer with sigmoid activation for binary classification

gru_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
gru_model.summary()
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is depr
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
gru (GRU)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

```
# Train the GRU model
gru_model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))
```

```
Epoch 1/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 134s 145ms/step - accuracy: 0.8660 - loss: 0.3002 - val_accuracy: 0.9291 - val_loss: 0.1864
Epoch 2/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 131s 145ms/step - accuracy: 0.9514 - loss: 0.1300 - val_accuracy: 0.9433 - val_loss: 0.1542
Epoch 3/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 131s 145ms/step - accuracy: 0.9673 - loss: 0.0924 - val_accuracy: 0.9402 - val_loss: 0.1598
Epoch 4/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 142s 145ms/step - accuracy: 0.9788 - loss: 0.0617 - val_accuracy: 0.9397 - val_loss: 0.1988
Epoch 5/5
902/902 ━━━━━━━━━━━━━━━━━━━━ 130s 144ms/step - accuracy: 0.9868 - loss: 0.0399 - val_accuracy: 0.9438 - val_loss: 0.2210
<keras.src.callbacks.history.History at 0x77fcd0a32f50>
```

```
# Evaluate the GRU model
gru_model.evaluate(X_test, y_test)
```

```
451/451 ━━━━━━━━━━━━━━━━━━━━ 10s 23ms/step - accuracy: 0.9430 - loss: 0.2185
[0.2210470587015152, 0.9438241124153137]
```