Lab 6 : RNN, LSTM & Transformers

1. Write a Python program to implement a text classification model using RNN and LSTM on the Twitter Sentiment140 dataset.

- Use pre-trained Word2Vec embeddings.
- Train both RNN and LSTM models.
- Compare their accuracy and training time.

```
#Importing required libraries
import pandas as pd
import re

dataset = pd.read_csv("twitter_sentiment.csv")
dataset.head()
```

	text	label
0	Feeling great today! Life is good.	1
1	Ugh, this weather sucks!	0
2	Can't wait for the weekend!	1
3	Traffic jam ruined my morning.	0
4	Loving the new phone!	1

Data Preprocessing

Some General Preprocessing and based on the assumptions for Twitter reviews. Rest have to check few records to find the pattern.

- Remove Null Values
- Remove punctuations
- Lower Case
- Remove digits and words containing digits
- Remove urls, mentions (@username) and topics (#viral)
- Remove stop words
- Lemmatization

```
# Display random sample of tweets
print(dataset.sample(10))
```

```
                                 text  label
44                 Sunshine and smiles!      1
48           Lovely dinner with family ❤️      1
27                 Missed my deadline 😓      0
45                  Got ignored all day.      0
24             Just finished a good book.      1
37        Too much work, no time to rest.      0
46  My favorite band released a new song!      1
7     Horrible service at the cafe today.      0
31         Spilled coffee all over myself.      0
30        Woke up early and feeling fresh!      1
```

```
# Display example tweets containing URLs, mentions, hashtags, emojis, etc.
sample_tweets = dataset[
    dataset['text'].str.contains(r"http|@|#", regex=True, na=False)
]

print(sample_tweets)
```

```
Empty DataFrame
Columns: [text, label]
Index: []
```

- There are no urls, mentions and hashtags
- But we have emojis

```
import nltk
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
from nltk.stem import WordNetLemmatizer
import emoji
```

```
!pip install emoji
```

```
Collecting emoji
  Downloading emoji-2.14.1-py3-none-any.whl.metadata (5.7 kB)
Downloading emoji-2.14.1-py3-none-any.whl (590 kB)
   ---------------------------------------- 0.0/590.6 kB ? eta -:--:--
   ----------------- ---------------------- 262.1/590.6 kB ? eta -:--:--
   ---------------------------------------- 590.6/590.6 kB 1.2 MB/s eta 0:00:00
Installing collected packages: emoji
Successfully installed emoji-2.14.1
```

```
# Download necessary NLTK resources (only once)
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
True
```

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def remove_emojis(text):
    return emoji.replace_emoji(text, replace='')

def preprocess_text(text):
    text = text.lower() #lowercase
    text = remove_emojis(text) # remove emoji
    text = re.sub(r'[^\w\s]', '', text) # remove punctuations
    text = re.sub(r'\b\w*\d\w*\b', '', text) #remove digits and words with digits
    words = word_tokenize(text)

    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]

    return ' '.join(words)

dataset['clean_text'] = dataset['text'].apply(preprocess_text)


(dataset.head(10))
```

	text	label	clean_text
0	Feeling great today! Life is good.	1	feeling great today life good
1	Ugh, this weather sucks!	0	ugh weather suck
2	Can't wait for the weekend!	1	cant wait weekend
3	Traffic jam ruined my morning.	0	traffic jam ruined morning
4	Loving the new phone!	1	loving new phone
5	Totally exhausted and frustrated.	0	totally exhausted frustrated
6	My dog just learned a new trick! So proud.	1	dog learned new trick proud
7	Horrible service at the cafe today.	0	horrible service cafe today
8	Had an amazing run. Endorphins are real!	1	amazing run endorphin real
9	Missed my flight. Worst day ever.	0	missed flight worst day ever

Use Pretrained Word2Vec Embedding

```
import gensim.downloader as api

print(list(api.info()['models'].keys()))
```

```
['fasttext-wiki-news-subwords-300', 'conceptnet-numberbatch-17-06-300', 'word2vec-ruscorpora-300', 'word2vec-google-news-300', 'glov
```

```
word2vec_model = api.load("glove-twitter-50")
```

```
[==================================================] 100.0% 199.5/199.5MB downloaded
```

```
max_len = dataset['clean_text'].str.len().max()
print(max_len)
```

```
33
```

```
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenize words
tokenizer = Tokenizer()
tokenizer.fit_on_texts(dataset['clean_text'])

# Convert text to sequences
sequences = tokenizer.texts_to_sequences(dataset['clean_text'])

# Set a fixed length for sequences
MAX_LEN = 30
padded_sequences = pad_sequences(sequences, maxlen=MAX_LEN, padding='post')

# Get vocabulary size
vocab_size = len(tokenizer.word_index) + 1
```

```
# Create Embedding Matrix

EMBEDDING_DIM = 50  #dim of glove-twitter-50

# Initialize an embedding matrix
embedding_matrix = np.zeros((vocab_size, EMBEDDING_DIM))

for word, i in tokenizer.word_index.items():
    if word in word2vec_model:
        embedding_matrix[i] = word2vec_model[word]
```

RNN Model

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

model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=EMBEDDING_DIM,
              weights=[embedding_matrix], trainable=False),
    SimpleRNN(128, activation='tanh', return_sequences=True),
    SimpleRNN(64, activation='tanh'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Model summary
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	?	6,200
simple_rnn_2 (SimpleRNN)	?	0 (unbuilt)
simple_rnn_3 (SimpleRNN)	?	0 (unbuilt)
dropout_1 (Dropout)	?	0
dense_1 (Dense)	?	0 (unbuilt)

Total params: 6,200 (24.22 KB)

```
from sklearn.model_selection import train_test_split
import time

X_train, X_test, y_train, y_test = train_test_split(padded_sequences, dataset['label'], test_size=0.2, random_state=42)

# Train model
start_time = time.time()
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
end_time = time.time()

training_time = end_time - start_time
print(f"Training Time: {training_time:.2f} seconds")
```

```
Epoch 1/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 859ms/step - accuracy: 0.5750 - loss: 0.7311 - val_accuracy: 0.7000 - val_loss: 0.5910
Epoch 2/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - accuracy: 0.6958 - loss: 0.6091 - val_accuracy: 0.9000 - val_loss: 0.5005
Epoch 3/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 218ms/step - accuracy: 0.7333 - loss: 0.5270 - val_accuracy: 0.8000 - val_loss: 0.4819
Epoch 4/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.8375 - loss: 0.4164 - val_accuracy: 0.7000 - val_loss: 0.5618
Epoch 5/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.9125 - loss: 0.3425 - val_accuracy: 0.7000 - val_loss: 0.6061
Epoch 6/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 0.9563 - loss: 0.2310 - val_accuracy: 0.7000 - val_loss: 0.5063
Epoch 7/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.9458 - loss: 0.2354 - val_accuracy: 0.8000 - val_loss: 0.4399
Epoch 8/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.9729 - loss: 0.1778 - val_accuracy: 0.9000 - val_loss: 0.3934
Epoch 9/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 190ms/step - accuracy: 0.9458 - loss: 0.1407 - val_accuracy: 0.9000 - val_loss: 0.3525
Epoch 10/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - accuracy: 0.9729 - loss: 0.0890 - val_accuracy: 0.8000 - val_loss: 0.3407
Training Time: 8.13 seconds
```

```
final_train_acc = history.history['accuracy'][-1]
final_val_acc = history.history['val_accuracy'][-1]

print(f"Final Training Accuracy: {final_train_acc:.4f}")
print(f"Final Validation Accuracy: {final_val_acc:.4f}")
```

```
Final Training Accuracy: 0.9750
Final Validation Accuracy: 0.8000
```

LSTM Model

```
from tensorflow.keras.layers import LSTM

# Define the LSTM model
lstm_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=50,
              weights=[embedding_matrix], trainable=False),
    LSTM(64, return_sequences=False),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# Compile the model
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Summary
lstm_model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	?	6,200
lstm (LSTM)	?	0 (unbuilt)
dropout_2 (Dropout)	?	0
dense_2 (Dense)	?	0 (unbuilt)

Total params: 6,200 (24.22 KB)

```
start_time = time.time()
lstm_history = lstm_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
end_time = time.time()

lstm_training_time = end_time - start_time
print(f"LSTM Training Time: {lstm_training_time:.2f} seconds")

final_lstm_train_acc = lstm_history.history['accuracy'][-1]
final_lstm_val_acc = lstm_history.history['val_accuracy'][-1]

print(f"LSTM Final Training Accuracy: {final_lstm_train_acc:.4f}")
print(f"LSTM Final Validation Accuracy: {final_lstm_val_acc:.4f}")
```

```
Epoch 1/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 5s 730ms/step - accuracy: 0.5437 - loss: 0.6932 - val_accuracy: 0.4000 - val_loss: 0.6934
Epoch 2/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.5333 - loss: 0.6925 - val_accuracy: 0.4000 - val_loss: 0.6940
Epoch 3/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - accuracy: 0.5479 - loss: 0.6910 - val_accuracy: 0.4000 - val_loss: 0.6943
Epoch 4/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step - accuracy: 0.5167 - loss: 0.6903 - val_accuracy: 0.4000 - val_loss: 0.6938
Epoch 5/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step - accuracy: 0.5271 - loss: 0.6889 - val_accuracy: 0.4000 - val_loss: 0.6927
Epoch 6/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.5062 - loss: 0.6859 - val_accuracy: 0.4000 - val_loss: 0.6904
Epoch 7/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - accuracy: 0.5708 - loss: 0.6769 - val_accuracy: 0.4000 - val_loss: 0.6864
Epoch 8/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 0.5646 - loss: 0.6702 - val_accuracy: 0.4000 - val_loss: 0.6777
Epoch 9/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.5604 - loss: 0.6538 - val_accuracy: 0.4000 - val_loss: 0.6597
Epoch 10/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 0.5979 - loss: 0.6042 - val_accuracy: 0.5000 - val_loss: 0.6351
LSTM Training Time: 6.70 seconds
LSTM Final Training Accuracy: 0.6000
LSTM Final Validation Accuracy: 0.5000
```

```
# Compare results
print(f"\nRNN Training Time: {training_time:.2f} sec vs LSTM Training Time: {lstm_training_time:.2f} sec")
print(f"RNN Validation Accuracy: {final_val_acc:.4f} vs LSTM Validation Accuracy: {final_lstm_val_acc:.4f}")
```

```

RNN Training Time: 8.13 sec vs LSTM Training Time: 6.70 sec
RNN Validation Accuracy: 0.8000 vs LSTM Validation Accuracy: 0.5000
```

2. Write a Python program using a pre-trained Transformer model (like BART or T5) from Hugging Face to perform text summarization.

- Accept a long article as input.
- Generate a concise summary.
- Evaluate using the different evaluation matrix scores.

```
!pip install torch transformers
```

```
      Found existing installation: nvidia-nvjitlink-cu12 12.5.82
      Uninstalling nvidia-nvjitlink-cu12-12.5.82:
        Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
    Attempting uninstall: nvidia-curand-cu12
      Found existing installation: nvidia-curand-cu12 10.3.6.82
      Uninstalling nvidia-curand-cu12-10.3.6.82:
        Successfully uninstalled nvidia-curand-cu12-10.3.6.82
    Attempting uninstall: nvidia-cufft-cu12
      Found existing installation: nvidia-cufft-cu12 11.2.3.61
      Uninstalling nvidia-cufft-cu12-11.2.3.61:
        Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
    Attempting uninstall: nvidia-cuda-runtime-cu12
      Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
      Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
    Attempting uninstall: nvidia-cuda-nvrtc-cu12
      Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
      Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
    Attempting uninstall: nvidia-cuda-cupti-cu12
      Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
      Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
    Attempting uninstall: nvidia-cublas-cu12
      Found existing installation: nvidia-cublas-cu12 12.5.3.2
      Uninstalling nvidia-cublas-cu12-12.5.3.2:
        Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
    Attempting uninstall: nvidia-cusparse-cu12
      Found existing installation: nvidia-cusparse-cu12 12.5.1.3
      Uninstalling nvidia-cusparse-cu12-12.5.1.3:
        Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
    Attempting uninstall: nvidia-cudnn-cu12
      Found existing installation: nvidia-cudnn-cu12 9.3.0.75
      Uninstalling nvidia-cudnn-cu12-9.3.0.75:
        Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
    Attempting uninstall: nvidia-cusolver-cu12
      Found existing installation: nvidia-cusolver-cu12 11.6.3.83
      Uninstalling nvidia-cusolver-cu12-11.6.3.83:
        Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
  Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-ru
```

```
import torch
from transformers import AutoTokenizer, AutoModelWithLMHead
```

```
# T5:  Text To Text Trasfer Transformer
tokenizer=AutoTokenizer.from_pretrained('T5-base')
model=AutoModelWithLMHead.from_pretrained('T5-base', return_dict=True)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning: 
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100% 1.21k/1.21k [00:00<00:00, 37.9kB/s]
spiece.model: 100% 792k/792k [00:00<00:00, 1.26MB/s]
tokenizer.json: 100% 1.39M/1.39M [00:00<00:00, 1.75MB/s]
/usr/local/lib/python3.11/dist-packages/transformers/models/auto/modeling_auto.py:1881: FutureWarning: The class `AutoModelWithLMHea
  warnings.warn(
model.safetensors: 100% 892M/892M [00:05<00:00, 167MB/s]
generation_config.json: 100% 147/147 [00:00<00:00, 13.4kB/s]
```

```
para = input("Enter a Para to be summarized : ")
```

```
Enter a Para to be summarized : Ad sales boost Time Warner profit  Quarterly profits at US media giant TimeWarner jumped 76% to $1.1
```

```
inputs=tokenizer.encode("sumarize: " +para,return_tensors='pt', max_length=512, truncation=True)

output = model.generate(inputs, min_length=80, max_length=200)

summary=tokenizer.decode(output[0])
```

```
import textwrap

# Wrap the summary text to a fixed width (e.g., 80 characters per line)
wrapped_summary = textwrap.fill(summary, width=80)
```

```
print(wrapped_summary)
```

```
<pad> time warner's profit jumped 76% to $1.13bn (£600m) for the three months to
December . firm benefited from sales of high-speed internet connections and
higher advert sales . time warner's film division saw profits slump 27% to $284m
. it is to restate its accounts as part of efforts to resolve an inquiry into
AOL .</s>
```

```
!pip install rouge-score
```

```
Collecting rouge-score
  Downloading rouge_score-0.1.2.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from rouge-score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (from rouge-score) (3.9.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from rouge-score) (2.0.2)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.11/dist-packages (from rouge-score) (1.17.0)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk->rouge-score) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk->rouge-score) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk->rouge-score) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk->rouge-score) (4.67.1)
Building wheels for collected packages: rouge-score
  Building wheel for rouge-score (setup.py) ... done
  Created wheel for rouge-score: filename=rouge_score-0.1.2-py3-none-any.whl size=24935 sha256=776758668c7c69580f5f83b9f96de2cd60cb
  Stored in directory: /root/.cache/pip/wheels/1e/19/43/8a442dc83660ca25e163e1bd1f89919284ab0d0c1475475148
Successfully built rouge-score
Installing collected packages: rouge-score
Successfully installed rouge-score-0.1.2
```

```
from rouge_score import rouge_scorer

def evaluate_summary(reference, generated):
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
    scores = scorer.score(reference, generated)
    return scores

reference_summary = input("Enter the Reference Summary: ")


rouge_scores = evaluate_summary(reference_summary, summary)

print("\nEvaluation Metrics:")
for key, value in rouge_scores.items():
    print(f"{key.upper()}: Precision={value.precision:.4f}, Recall={value.recall:.4f}, F1-score={value.fmeasure:.4f}")
```

```
Enter the Reference Summary: TimeWarner said fourth quarter sales rose 2% to $11.1bn from $10.9bn.For the full-year, TimeWarner pos

Evaluation Metrics:
ROUGE1: Precision=0.5690, Recall=0.2157, F1-score=0.3128
ROUGE2: Precision=0.2456, Recall=0.0921, F1-score=0.1340
ROUGEL: Precision=0.3793, Recall=0.1438, F1-score=0.2085
```