## EX 9 TEXT GENERATION USING LSTM NETWORKS

### Problem Statement:

Build a text generation model using Long Short-Term Memory (LSTM) networks. Train the model on a text corpus to generate coherent sequences of text and evaluate the output for fluency and coherence.

Suggested Dataset: Shakespeare Corpus

### Objectives:

1. Understand sequential modeling for natural language generation.
2. Train a character-level LSTM model to learn language patterns.
3. Generate text using a seed prompt and evaluate the results.
4. Analyze the fluency and creativity of LSTM-generated outputs.

### Scope:

Text generation is a foundational task in natural language processing. This experiment demonstrates how LSTMs can learn syntactic and semantic patterns over time and generate believable sequences of text. The use of character-level modeling helps capture detailed language structures.

Tools and Libraries Used:

1. Python 3.x
2. TensorFlow / Keras
3. NumPy
4. Shakespeare Text Corpus (Tiny Shakespeare)

### Implementation Steps:

### Step 1: Load and Preprocess the Dataset

```
import tensorflow as tf
import numpy as np

text = tf.keras.utils.get_file('shakespeare.txt',
   'https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt')
text = open(text, 'r').read().lower()
chars = sorted(set(text))
c2i = {c: i for i, c in enumerate(chars)}
i2c = {i: c for i, c in enumerate(chars)}
```

### Step 2: Create Input and Output Sequences

```
seq_len = 40
X = []
y = []

for i in range(len(text) - seq_len):
   input_seq = text[i:i + seq_len]
   target_char = text[i + seq_len]
   X.append([c2i[c] for c in input_seq])
   y.append(c2i[target_char])

X = np.array(X)
y = np.array(y)
```

### Step 3: Build the LSTM Model

```
model = tf.keras.Sequential([
   tf.keras.layers.Embedding(len(chars), 64, input_length=seq_len),
   tf.keras.layers.LSTM(128),
   tf.keras.layers.Dense(len(chars), activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')
model.fit(X, y, batch_size=128, epochs=1)
```

### Step 4: Define the Text Generation Function

```
def generate(seed, length=300):
   seq = [c2i[c] for c in seed.lower()]
   for _ in range(length):
      inp = np.array(seq[-seq_len:]).reshape(1, -1)
      pred = model.predict(inp, verbose=0)[0]
      next_idx = np.random.choice(len(pred), p=pred)
      seq.append(next_idx)
   return seed + ''.join(i2c[i] for i in seq[len(seed):])
```

### Step 5: Generate and Display Text

```
print("\nGenerated Text:\n")
print(generate("shall i compare thee to a summer's day?\n"))
```

**Output:**

```
Generated Text:

shall i compare thee to a summer's day?

kind worldmbly:
be the was of before spyech will of beopker, i ayf.

lucendeo:
that what would thim anst marbned unto you.

vicinent:
the fyore!

duke intimnes:
we'se sither, and immalio,
foil i for is of autenel, go but, i deas
them our lieg. ruclio?
our to younce a face and poling,
and this the h
```