EXP NO: 09 DATE: 04/10/2025	**Text Generation using LSTM**

AIM: To train an LSTM-based recurrent neural network on the Shakespeare corpus to generate coherent and fluent English-like text sequences.

ALGORITHM:

- Load and preprocess the Shakespeare dataset (convert to lowercase, tokenize characters).
- Create input sequences of fixed length for training (each sequence predicts the next character).
- Build a Sequential LSTM model with embedding and dense output layers.
- Compile the model with categorical cross-entropy loss and Adam optimizer.
- Train the model on text sequences for several epochs.
- Generate new text by seeding the model with a random starting string and predicting next characters iteratively.

CODE:

```
# Text Generation with LSTM (Shakespeare Corpus)

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Embedding

import numpy as np

import random, sys


# Load dataset

path = tf.keras.utils.get_file("shakespeare.txt",

   "https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.txt")

text = open(path, "r", encoding="utf-8").read().lower()

print(f"Corpus length: {len(text)} characters")
```

```
# Create character mappings
chars = sorted(list(set(text)))
char2idx = {c:i for i, c in enumerate(chars)}
idx2char = {i:c for i, c in enumerate(chars)}

# Prepare sequences
seq_len = 60
step = 3
sentences = []
next_chars = []
for i in range(0, len(text) - seq_len, step):
    sentences.append(text[i: i + seq_len])
    next_chars.append(text[i + seq_len])
print("Number of sequences:", len(sentences))

x = np.zeros((len(sentences), seq_len, len(chars)), dtype=bool)
y = np.zeros((len(sentences), len(chars)), dtype=bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        x[i, t, char2idx[char]] = 1
    y[i, char2idx[next_chars[i]]] = 1

# Build model
model = Sequential([
    LSTM(128, input_shape=(seq_len, len(chars))),
    Dense(len(chars), activation='softmax')
])
model.compile(loss='categorical_crossentropy',  optimizer='adam')
```

```
model.fit(x, y, batch_size=128, epochs=20)


# Function to sample next character
def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds + 1e-8) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)


# Generate text
start_index = random.randint(0, len(text) - seq_len - 1)
seed_text = text[start_index:start_index + seq_len]
print("Seed:\n", seed_text)
print("\nGenerated Text:\n")


generated = seed_text
for i in range(500):
    x_pred = np.zeros((1, seq_len, len(chars)))
    for t, char in enumerate(seed_text):
        x_pred[0, t, char2idx[char]] = 1
    preds = model.predict(x_pred, verbose=0)[0]
    next_index = sample(preds, temperature=0.5)
    next_char = idx2char[next_index]
    generated += next_char
    seed_text = seed_text[1:] + next_char
```

print(generated)

OUTPUT:

```
romeo: but that kill my heart!

king henry vi:
why, are you so brief?

second murderer:
soft! was ever man so wontmen!

benvolio:
tut, then, i hope, sir, my mistaking sorrow on the sight.

juliet:
o, sir, your cartisfy!

ablisti:
let them call upon you alive,
who in a birthmen to marry warwick as meet,
to
```

RESULT: The LSTM model successfully learned character-level language structure and generated Shakespeare-like text with realistic word formations and dialogue styles.