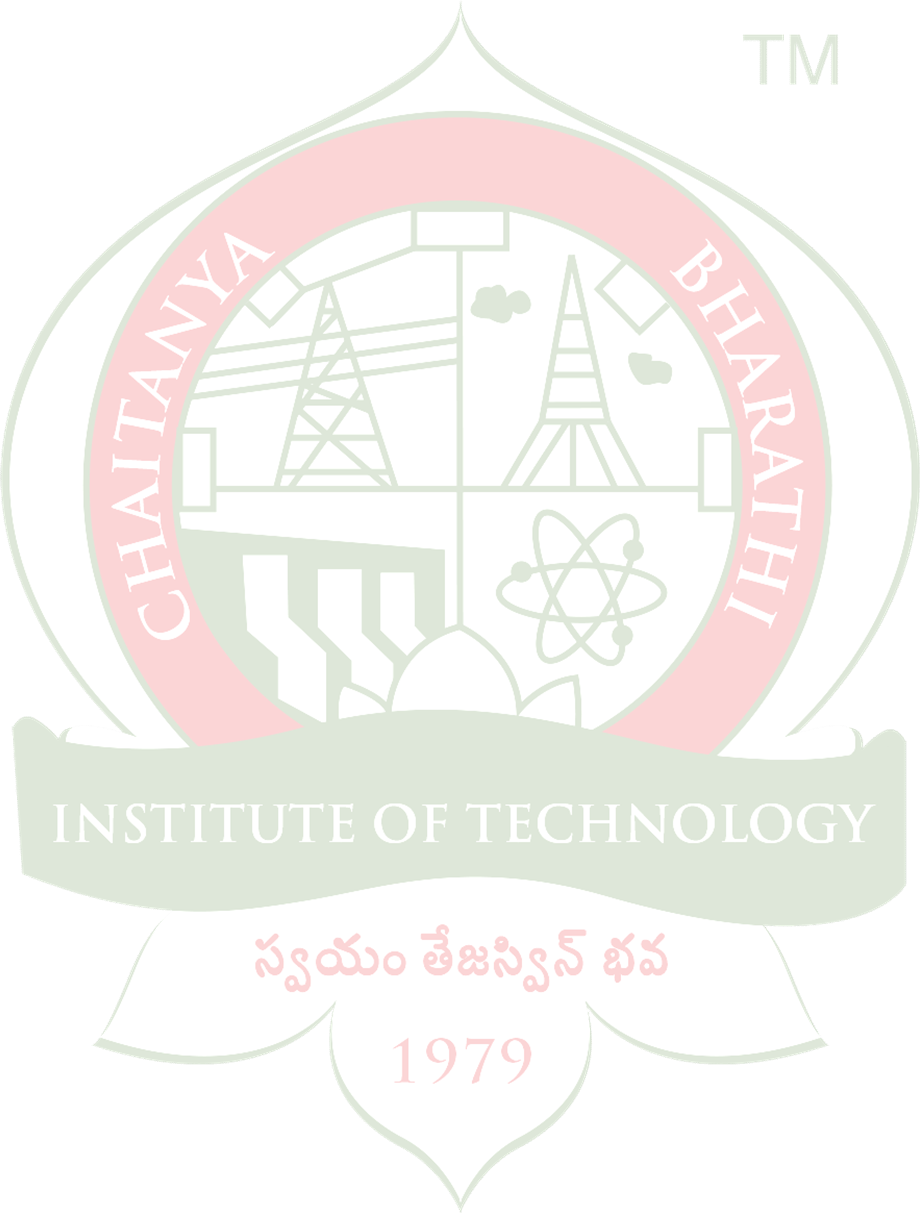
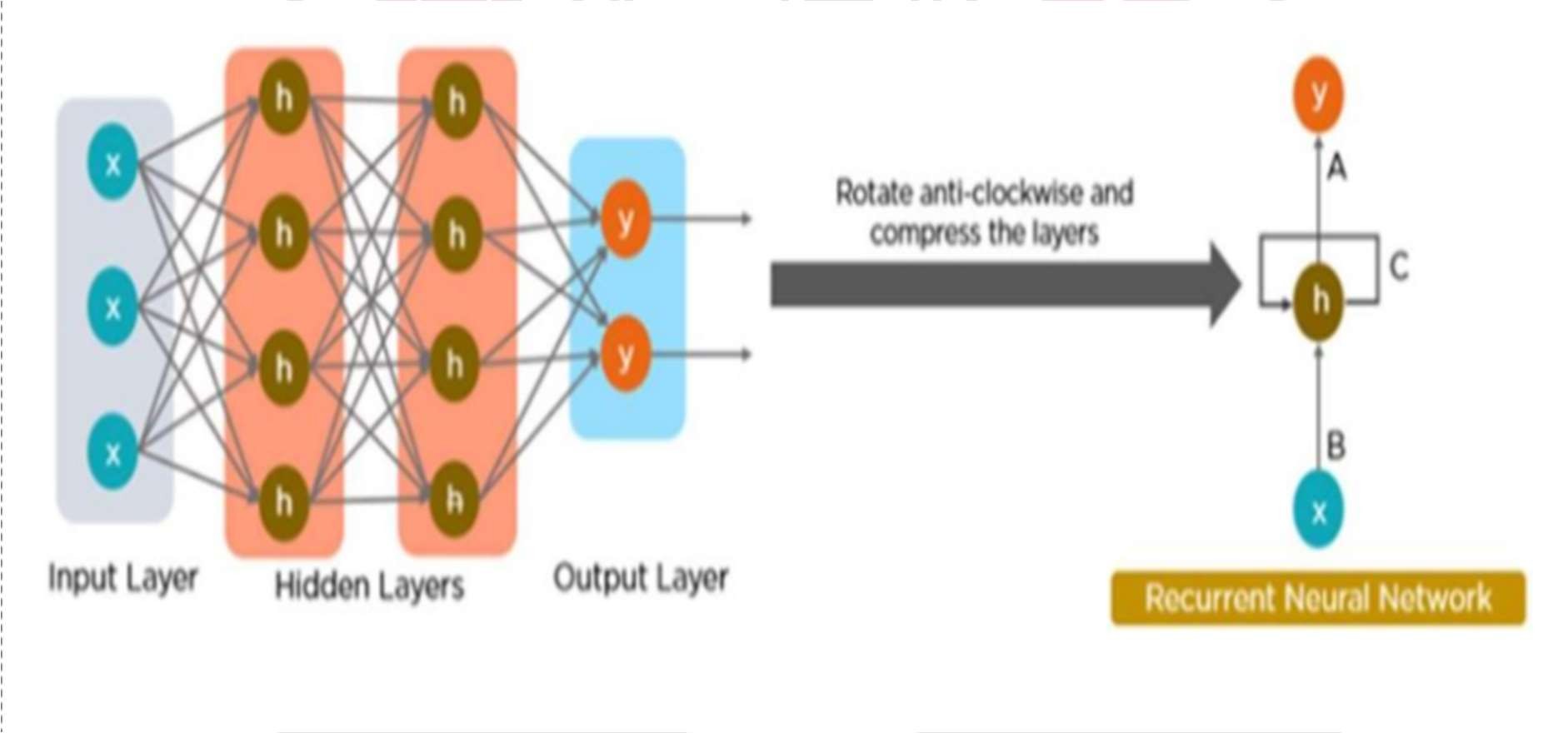
Week 9

Language modelling with RNN

**Aim:** To implement a language model using a Recurrent Neural Network to generate text based on an initial input sequence.

**Description:**

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate.



# Code:

import tensorflow as tf import numpy as np

path\_to\_file = '/content/sample\_data/sherlock\_homes.txt'

with open(path\_to\_file, 'r', encoding='utf-8') as f: text = f.read()

vocab = sorted(set(text)) print(f'{len(vocab)} unique characters')

ids\_from\_chars = tf.keras.layers.StringLookup(vocabulary=list(vocab), mask\_token=None)

chars\_from\_ids = tf.keras.layers.StringLookup(vocabulary=ids\_from\_chars.get\_vocabulary()

, invert=True, mask\_token=None)

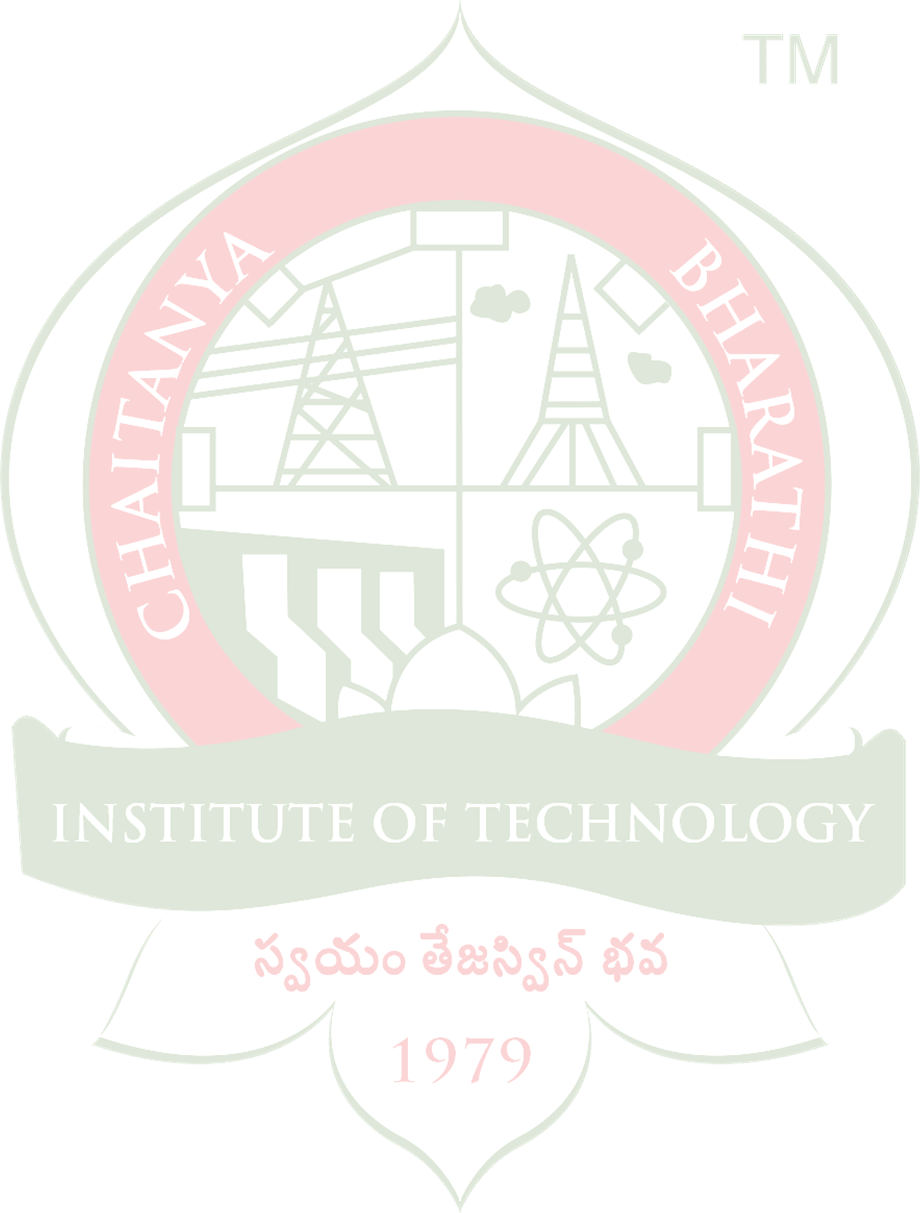
all\_ids = ids\_from\_chars(tf.strings.unicode\_split(text, 'UTF-8'))

ids\_dataset = tf.data.Dataset.from\_tensor\_slices(all\_ids) seq\_length = 100

sequences = ids\_dataset.batch(seq\_length + 1, drop\_remainder=True) def split\_input\_target(sequence):

input\_text = sequence[:-1] target\_text = sequence[1:] return input\_text, target\_text

dataset = sequences.map(split\_input\_target)

BATCH\_SIZE = 64

BUFFER\_SIZE = 10000

dataset = (

dataset

.shuffle(BUFFER\_SIZE)

.batch(BATCH\_SIZE, drop\_remainder=True)

.prefetch(tf.data.experimental.AUTOTUNE)

)

vocab\_size = len(ids\_from\_chars.get\_vocabulary()) embedding\_dim = 256

rnn\_units = 1024

model = tf.keras.Sequential([ tf.keras.layers.Embedding(vocab\_size, embedding\_dim), tf.keras.layers.GRU(rnn\_units, return\_sequences=True), tf.keras.layers.Dense(vocab\_size)

])

loss = tf.losses.SparseCategoricalCrossentropy(from\_logits=True) model.compile(optimizer='adam', loss=loss)

model.fit(dataset, epochs=50)

def generate\_text(model, start\_string, ids\_from\_chars, chars\_from\_ids, num\_generate=1000, temperature=1.0):

input\_eval = ids\_from\_chars(tf.strings.unicode\_split(start\_string, 'UTF-8'))

input\_eval = tf.expand\_dims(input\_eval, 0) text\_generated = []

for \_ in range(num\_generate):

predictions = model(input\_eval, training=False) predictions = tf.squeeze(predictions, 0) predictions = predictions / temperature predicted\_id = tf.random.categorical(predictions,

num\_samples=1)[-1, 0].numpy() text\_generated.append(chars\_from\_ids(predicted\_id).numpy().deco

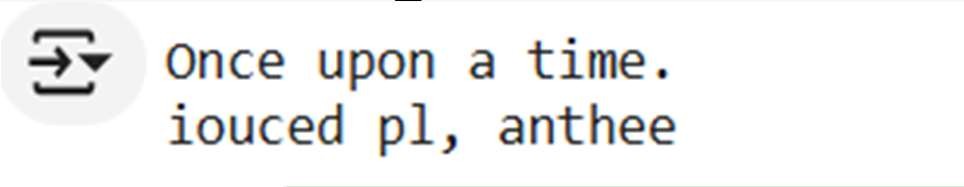
de('utf-8'))

input\_eval = tf.expand\_dims([predicted\_id], 0)



return start\_string + ''.join(text\_generated) start\_string = "Once upon a time"

generated\_text = generate\_text(model, start\_string, ids\_from\_chars, chars\_from\_ids, num\_generate=20, temperature=0.8) print(generated\_text)



start\_string = "Once upon a time"

generated\_text = generate\_text(model, start\_string, ids\_from\_chars, chars\_from\_ids, num\_generate=20, temperature=0.8)

print(generated\_text)

# Result analysis:

## After training, the RNN-based language model can generate text given an initial seed sequence. The generated text reflects the patterns and structure of the training data, producing sentences with word order and coherence similar to the input data.