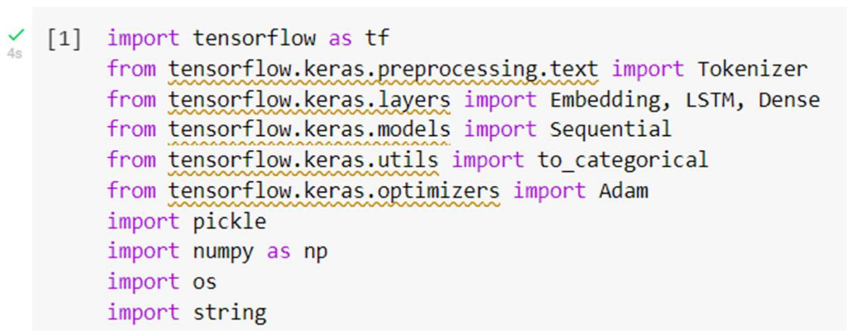
Topics in Deep learning Hands-On Unit 3

Name – B Pravena Section – B

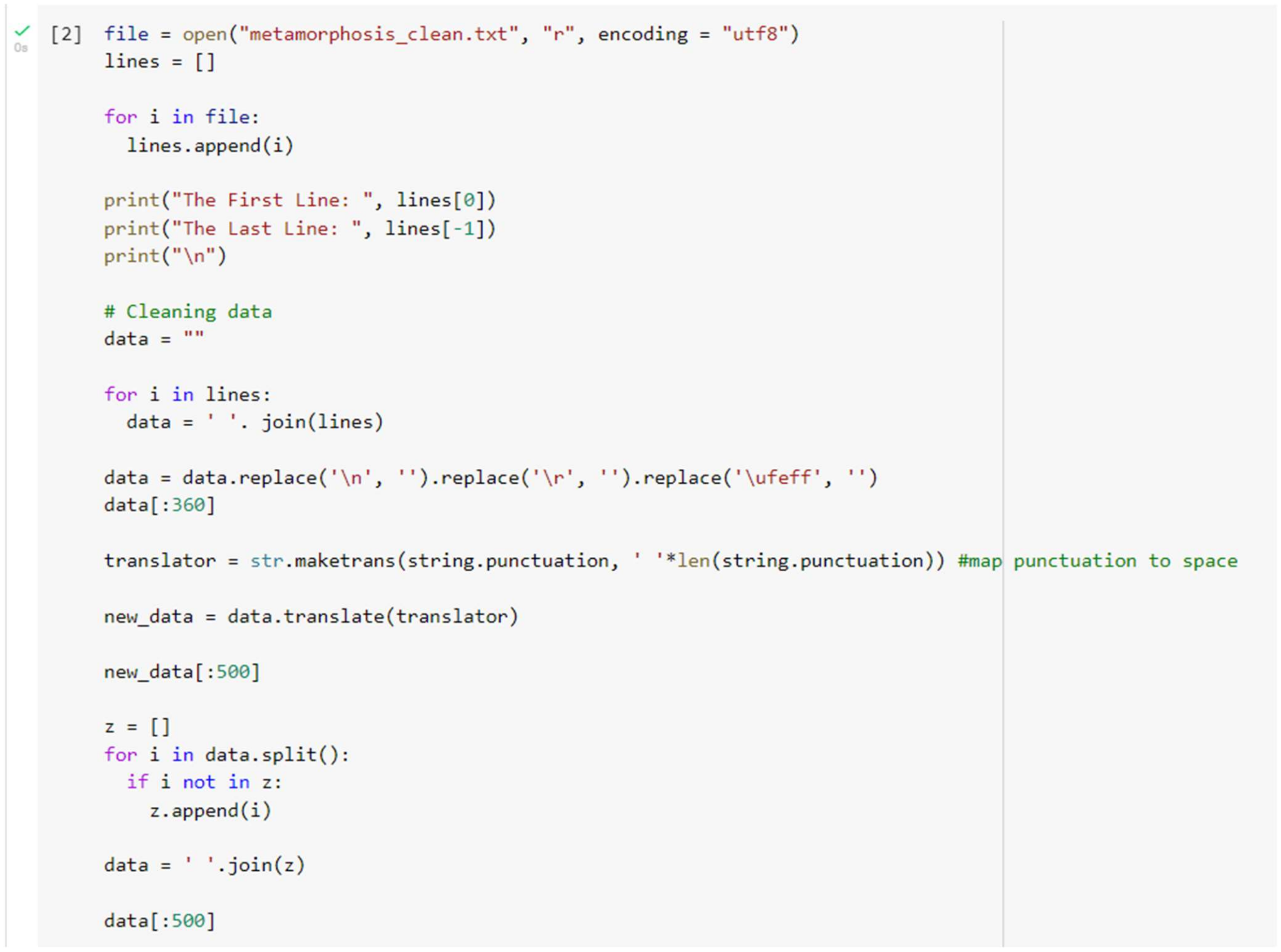
SRN – PES2UG19CS076

Google collab link -:

https://colab.research.google.com/drive/1KGu-SsmmU0yr99UUmBlkK8mmkYUMaYJW

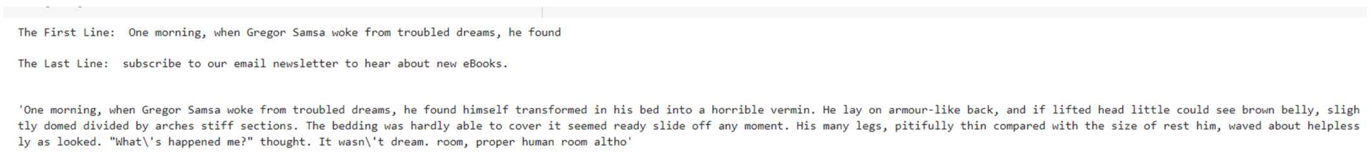


```
[1]  import tensorflow as tf
     from tensorflow.keras.preprocessing.text import Tokenizer
     from tensorflow.keras.layers import Embedding, LSTM, Dense
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.utils import to_categorical
     from tensorflow.keras.optimizers import Adam
     import pickle
     import numpy as np
     import os
     import string
```



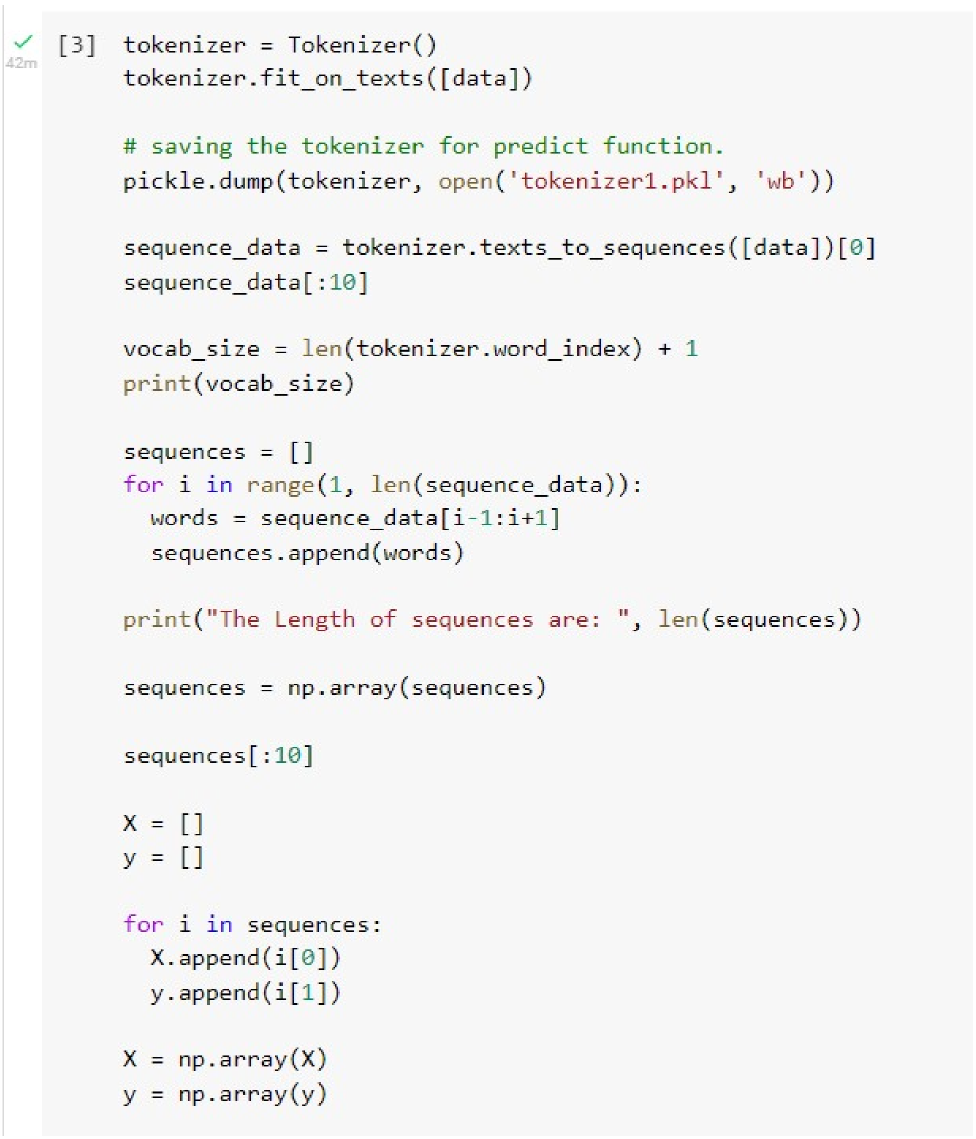
```
[2]  file = open("metamorphosis_clean.txt", "r", encoding = "utf8")
     lines = []

     for i in file:
       lines.append(i)

     print("The First Line: ", lines[0])
     print("The Last Line: ", lines[-1])
     print("\n")

     # Cleaning data
     data = ""

     for i in lines:
       data = ' '. join(lines)

     data = data.replace('\n', '').replace('\r', '').replace('\ufeff', '')
     data[:360]

     translator = str.maketrans(string.punctuation, ' '*len(string.punctuation)) #map punctuation to space

     new_data = data.translate(translator)

     new_data[:500]

     z = []
     for i in data.split():
       if i not in z:
         z.append(i)

     data = ' '.join(z)

     data[:500]
```

Output -:



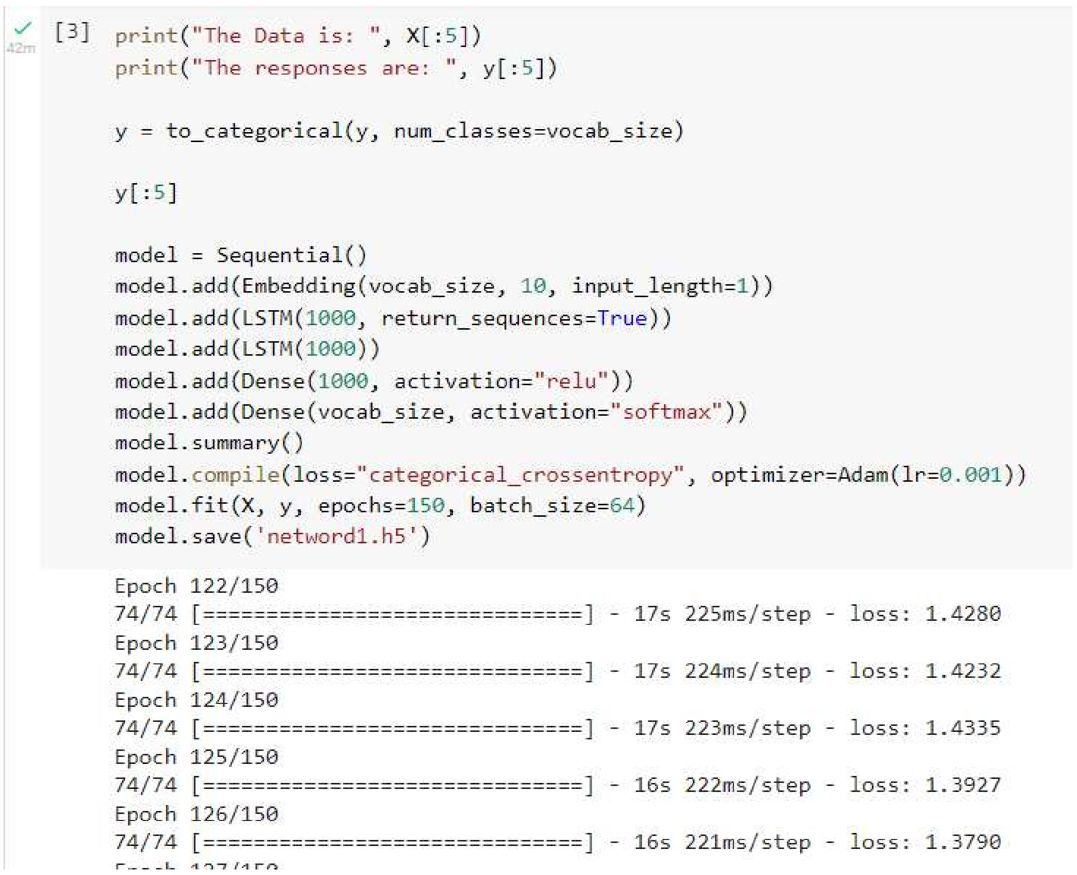
```
The First Line:  One morning, when Gregor Samsa woke from troubled dreams, he found

The Last Line:  subscribe to our email newsletter to hear about new eBooks.


'One morning, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin. He lay on armour-like back, and if lifted head little could see brown belly, sligh
tly domed divided by arches stiff sections. The bedding was hardly able to cover it seemed ready slide off any moment. His many legs, pitifully thin compared with the size of rest him, waved about helpless
ly as looked. "What\'s happened me?" thought. It wasn\'t dream. room, proper human room altho'
```

Tokenization -:



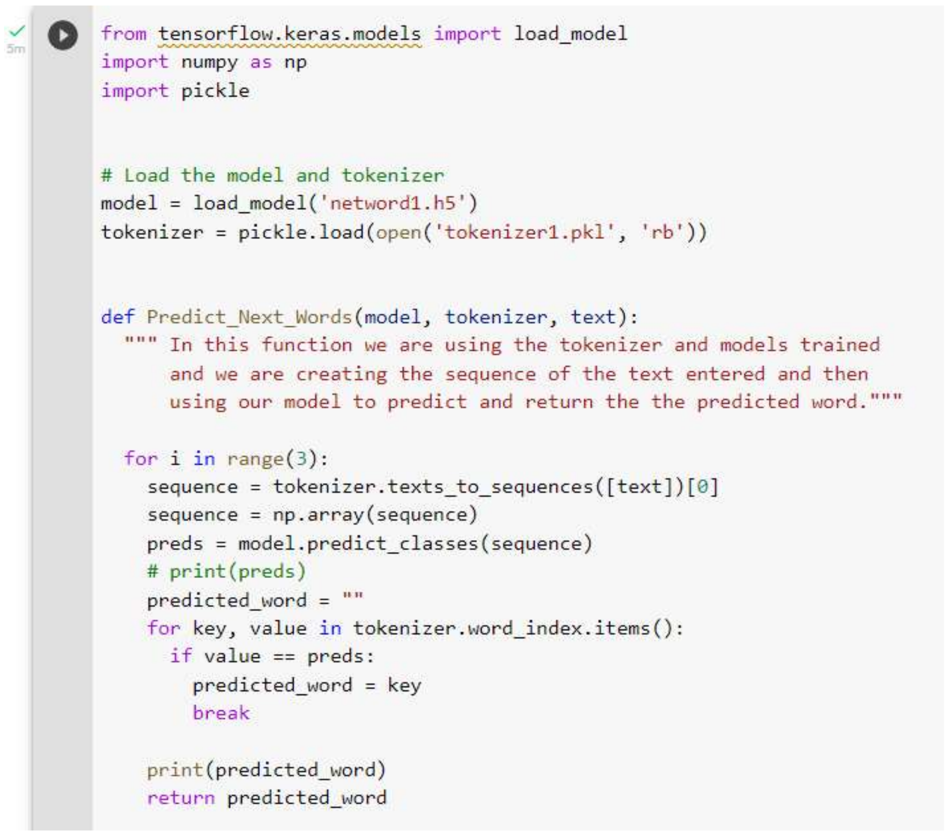
```
[3]
42m
```

```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts([data])

# saving the tokenizer for predict function.
pickle.dump(tokenizer, open('tokenizer1.pkl', 'wb'))

sequence_data = tokenizer.texts_to_sequences([data])[0]
sequence_data[:10]

vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)

sequences = []
for i in range(1, len(sequence_data)):
  words = sequence_data[i-1:i+1]
  sequences.append(words)

print("The Length of sequences are: ", len(sequences))

sequences = np.array(sequences)

sequences[:10]

X = []
y = []

for i in sequences:
  X.append(i[0])
  y.append(i[1])

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

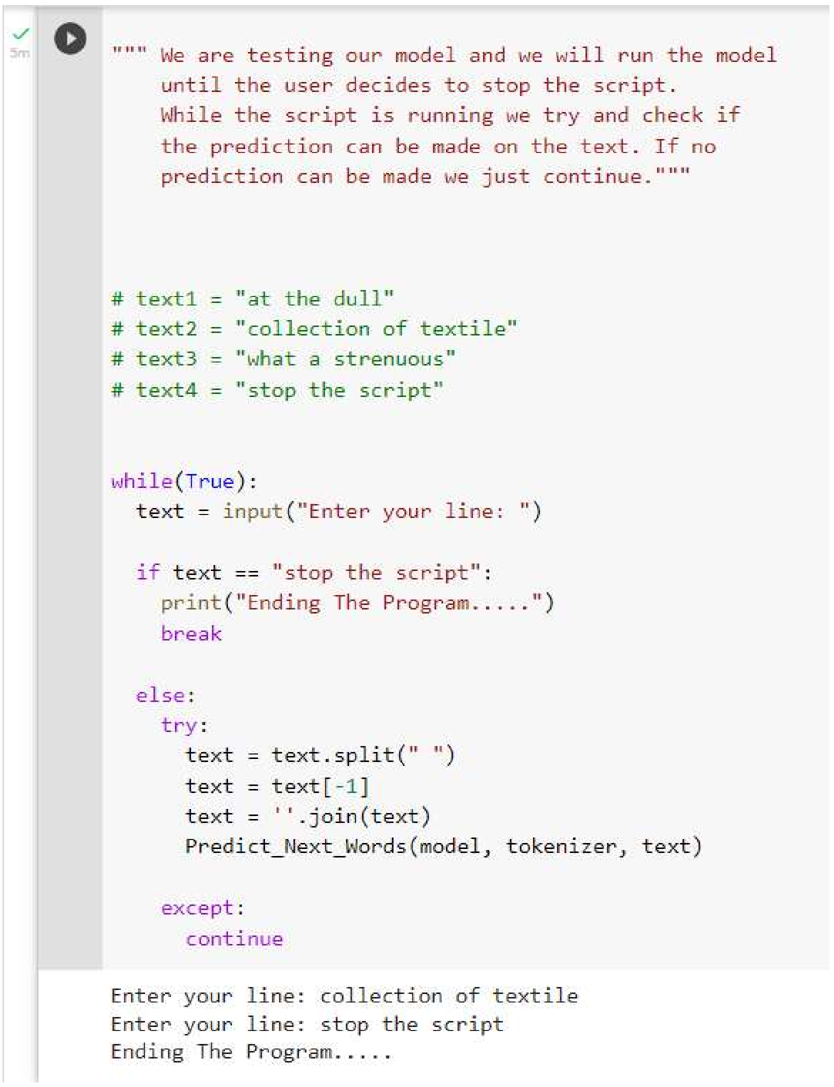


```
[3]  print("The Data is: ", X[:5])
     print("The responses are: ", y[:5])

     y = to_categorical(y, num_classes=vocab_size)

     y[:5]

     model = Sequential()
     model.add(Embedding(vocab_size, 10, input_length=1))
     model.add(LSTM(1000, return_sequences=True))
     model.add(LSTM(1000))
     model.add(Dense(1000, activation="relu"))
     model.add(Dense(vocab_size, activation="softmax"))
     model.summary()
     model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.001))
     model.fit(X, y, epochs=150, batch_size=64)
     model.save('netword1.h5')
```

```
Epoch 122/150
74/74 [==============================] - 17s 225ms/step - loss: 1.4280
Epoch 123/150
74/74 [==============================] - 17s 224ms/step - loss: 1.4232
Epoch 124/150
74/74 [==============================] - 17s 223ms/step - loss: 1.4335
Epoch 125/150
74/74 [==============================] - 16s 222ms/step - loss: 1.3927
Epoch 126/150
74/74 [==============================] - 16s 221ms/step - loss: 1.3790
```



```
from tensorflow.keras.models import load_model
import numpy as np
import pickle


# Load the model and tokenizer
model = load_model('netword1.h5')
tokenizer = pickle.load(open('tokenizer1.pkl', 'rb'))


def Predict_Next_Words(model, tokenizer, text):
  """ In this function we are using the tokenizer and models trained
      and we are creating the sequence of the text entered and then
      using our model to predict and return the the predicted word."""

  for i in range(3):
    sequence = tokenizer.texts_to_sequences([text])[0]
    sequence = np.array(sequence)
    preds = model.predict_classes(sequence)
    # print(preds)
    predicted_word = ""
    for key, value in tokenizer.word_index.items():
      if value == preds:
        predicted_word = key
        break

    print(predicted_word)
    return predicted_word
```



```python
""" We are testing our model and we will run the model
    until the user decides to stop the script.
    While the script is running we try and check if
    the prediction can be made on the text. If no
    prediction can be made we just continue."""



# text1 = "at the dull"
# text2 = "collection of textile"
# text3 = "what a strenuous"
# text4 = "stop the script"


while(True):
  text = input("Enter your line: ")

  if text == "stop the script":
    print("Ending The Program.....")
    break

  else:
    try:
      text = text.split(" ")
      text = text[-1]
      text = ''.join(text)
      Predict_Next_Words(model, tokenizer, text)

    except:
      continue
```

```
Enter your line: collection of textile
Enter your line: stop the script
Ending The Program.....
```