20BCE057 DEVASY PATEL

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
····for·filename·in·filenames:
········print(os.path.join(dirname,·filename))
```

```
from tensorflow import keras
import numpy
from numpy import array
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Embedding
#from keras.preprocessing.sequence import pad_sequences

data = """ I am a student and my name is devasy .\n I love to cycle.\n i studied in nirma university .\n My sister name is priyal ."""
print("Data:", data, type(data))

data_splitted=data.split('\n') #returns a list of strings
print("Data_Splitted:", data_splitted, type(data_splitted))
```

```
Data:  I am a student and my name is devasy .
 I love to cycle.
 i studied in nirma university .
 My sister name is priyal . <class 'str'>
Data_Splitted: [' I am a student and my name is devasy .', ' I love to cycle.', ' i studied in nirma university .', ' My sister
```

```
tokenizer=Tokenizer(filters='!"#$%&()*+,-/:;<=>?@[\]^_`{|}~')

tokenizer.fit_on_texts(data_splitted) #learns a vocabulary
print("Word Indices:", tokenizer.word_index) #tokenizer.word_index is a dictionary
```

```
Word Indices: {'i': 1, '.': 2, 'my': 3, 'name': 4, 'is': 5, 'am': 6, 'a': 7, 'student': 8, 'and': 9, 'devasy': 10, 'love': 11, 'to':
```

```
vocab_size = len(tokenizer.word_index) + 1
print("Vocab Size:", vocab_size)

sequences=tokenizer.texts_to_sequences(data_splitted) #list of list
l=len(sequences)
print("Sequences:",sequences, type(sequences), l)
```

```
Vocab Size: 21
Sequences: [[1, 6, 7, 8, 9, 3, 4, 5, 10, 2], [1, 11, 12, 13, 14], [1, 15, 16, 17, 18, 2], [3, 19, 4, 5, 20, 2]] <class 'list'> 4
```

```
X=list()
y=list()

for i in range(len(sequences)):
  X.insert(i,sequences[i][:-1])
  y.insert(i,sequences[i])

print("X=", X, "y=",y, type(X), type(y))

#In x we remove all the last word
```

```
X= [[1, 6, 7, 8, 9, 3, 4, 5, 10], [1, 11, 12, 13], [1, 15, 16, 17, 18], [3, 19, 4, 5, 20]] y= [[1, 6, 7, 8, 9, 3, 4, 5, 10, 2], [1,
```

```
maxlen = max([len(sequence) for sequence in X])
print("Maxlen:",maxlen)

# First sentense in X
```

```
Maxlen: 9
```

```
from keras.preprocessing.sequence import pad_sequences
from keras.utils import pad_sequences


X=pad_sequences(X,maxlen=maxlen+1,padding='pre') # +1 to have 0 as the first input
print("X:",X, type(X), X.shape)
y=pad_sequences(y,maxlen=maxlen+1,padding='pre')
print("y:",y, type(y), y.shape)
```

```
X: [[ 0  1  6  7  8  9  3  4  5 10]
 [ 0  0  0  0  0  0  1 11 12 13]
 [ 0  0  0  0  0  1 15 16 17 18]
 [ 0  0  0  0  0  3 19  4  5 20]] <class 'numpy.ndarray'> (4, 10)
y: [[ 1  6  7  8  9  3  4  5 10  2]
 [ 0  0  0  0  0  1 11 12 13 14]
 [ 0  0  0  0  1 15 16 17 18  2]
 [ 0  0  0  0  3 19  4  5 20  2]] <class 'numpy.ndarray'> (4, 10)
```

```
print("X=", X, "y=",y, type(X), type(y), X.shape, y.shape)
model=Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=21))
print(model.output_shape)
```

```
X= [[ 0  1  6  7  8  9  3  4  5 10]
    [ 0  0  0  0  0  0  1 11 12 13]
    [ 0  0  0  0  0  1 15 16 17 18]
    [ 0  0  0  0  0  3 19  4  5 20]] y= [[ 1  6  7  8  9  3  4  5 10  2]
    [ 0  0  0  0  0  1 11 12 13 14]
    [ 0  0  0  0  1 15 16 17 18  2]
    [ 0  0  0  0  3 19  4  5 20  2]] <class 'numpy.ndarray'> <class 'numpy.ndarray'> (4, 10) (4, 10)
   (None, None, 21)
```

```
model.add(SimpleRNN(units=50, return_sequences=True))
print(model.output_shape)
```

```
(None, None, 50)
```

```
model.add(Dense(units=vocab_size,activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 21)          441

 simple_rnn_1 (SimpleRNN)    (None, None, 50)          3600

 dense_1 (Dense)             (None, None, 21)          1071

=================================================================
Total params: 5,112
Trainable params: 5,112
Non-trainable params: 0
_________________________________________________________________
```

```
print("Input Shape of all Layers:",model.layers[0].input_shape,model.layers[1].input_shape,model.layers[2].input_shape)
print("Input Dim:",model.layers[0].input_dim)
```

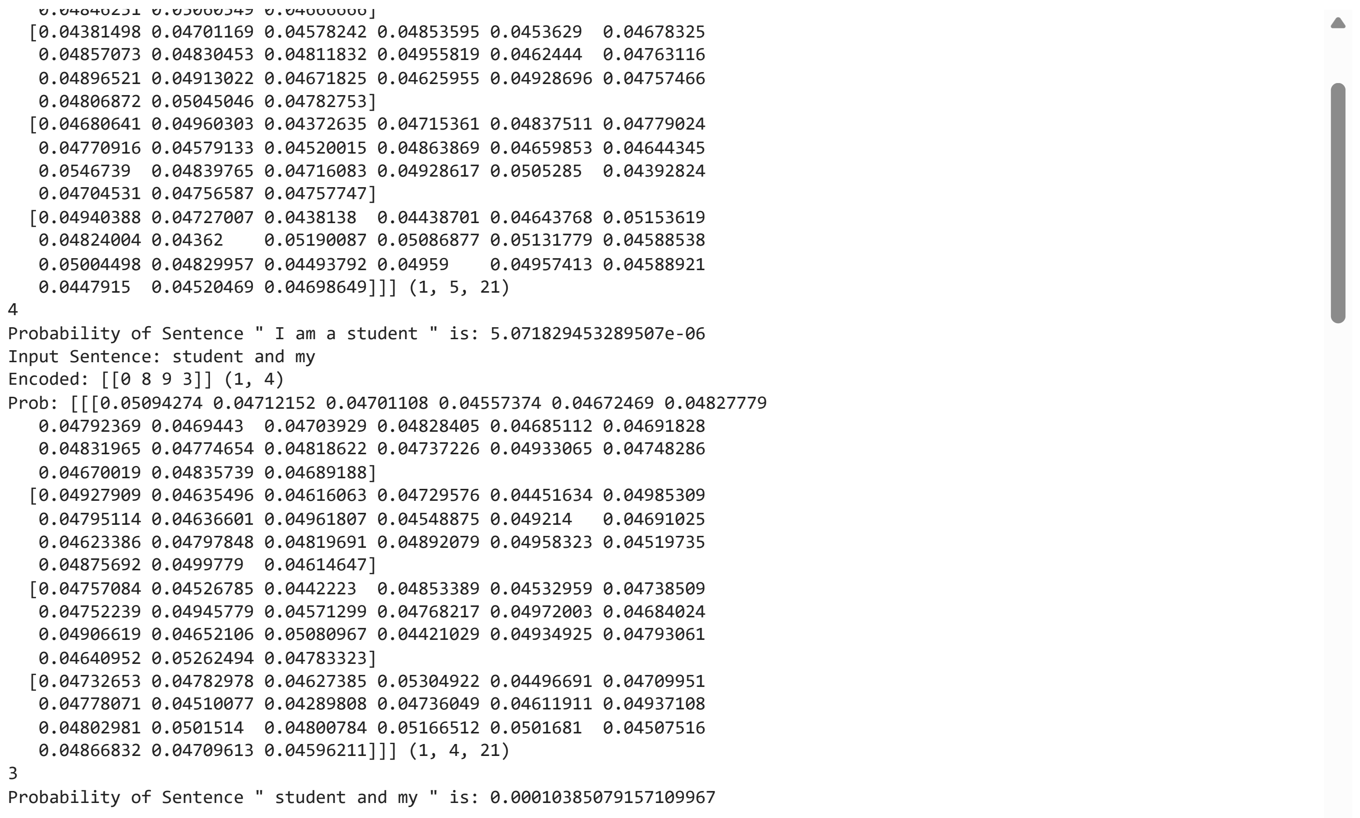
```
Input Shape of all Layers: (None, None) (None, None, 21) (None, None, 50)
Input Dim: 21
```

```
def prob_of_input_sentence(model, tokenizer, sentence):
        print("Input Sentence:", sentence)
        encoded=tokenizer.texts_to_sequences([sentence])[0]
        encoded.insert(0,0)
        encoded=array(encoded)
        encoded=numpy.reshape(encoded,newshape=(1,-1))
        print("Encoded:", encoded, encoded.shape)
        prob=model.predict(encoded, verbose=0)
        print("Prob:", prob, prob.shape)
        probability=1
        print(prob.shape[1]-1)
        for i in range(prob.shape[1]-1):
              probability = probability * prob[0,i,encoded[0,i+1]]
        print("Probability of Sentence", "\"", sentence, "\"", "is:", probability)


print("------------------------------------------------------------")
prob_of_input_sentence(model, tokenizer, "I am a student")
# I am a student and my name is devasy .
#I love to cycle.
#i studied in nirma university .\n My sister name is priyal ."""

prob_of_input_sentence(model, tokenizer, "student and my")
prob_of_input_sentence(model, tokenizer, "my name is devasy")
prob_of_input_sentence(model, tokenizer, "i am a devasy .")
prob_of_input_sentence(model, tokenizer, "i love to cycle")
prob_of_input_sentence(model, tokenizer, "love to read")
prob_of_input_sentence(model, tokenizer, "to cycle")
prob_of_input_sentence(model, tokenizer, "studied in nirma")
prob_of_input_sentence(model, tokenizer, "i studied in nirma")
prob_of_input_sentence(model, tokenizer, "in nirma university")
prob_of_input_sentence(model, tokenizer, "i studied in nirma university")
prob_of_input_sentence(model, tokenizer, "My sister name")
prob_of_input_sentence(model, tokenizer, "name is priyal")
prob_of_input_sentence(model, tokenizer, "My sister name is priyal.")
print("------------------------------------------------------------------------------")

#Word Indices: {'i': 1, '.': 2, 'my': 3, 'name': 4, 'is': 5, 'am': 6, 'a': 7, 'student': 8,
#'and': 9, 'devasy': 10, 'love': 11, 'to': 12, 'read': 13, 'books.': 14, 'studied': 15,
#'in': 16, 'nirma': 17, 'university': 18, 'sister': 19, 'priyal': 20}
```



```
   0.04846251 0.05060549 0.04666666]
  [0.04381498 0.04701169 0.04578242 0.04853595 0.0453629  0.04678325
   0.04857073 0.04830453 0.04811832 0.04955819 0.0462444  0.04763116
   0.04896521 0.04913022 0.04671825 0.04625955 0.04928696 0.04757466
   0.04806872 0.05045046 0.04782753]
  [0.04680641 0.04960303 0.04372635 0.04715361 0.04837511 0.04779024
   0.04770916 0.04579133 0.04520015 0.04863869 0.04659853 0.04644345
   0.0546739  0.04839765 0.04716083 0.04928617 0.0505285  0.04392824
   0.04704531 0.04756587 0.04757747]
  [0.04940388 0.04727007 0.0438138  0.04438701 0.04643768 0.05153619
   0.04824004 0.04362    0.05190087 0.05086877 0.05131779 0.04588538
   0.05004498 0.04829957 0.04493792 0.04959    0.04957413 0.04588921
   0.0447915  0.04520469 0.04698649]]] (1, 5, 21)
4
Probability of Sentence " I am a student " is: 5.071829453289507e-06
Input Sentence: student and my
Encoded: [[0 8 9 3]] (1, 4)
Prob: [[[0.05094274 0.04712152 0.04701108 0.04557374 0.04672469 0.04827779
   0.04792369 0.0469443  0.04703929 0.04828405 0.04685112 0.04691828
   0.04831965 0.04774654 0.04818622 0.04737226 0.04933065 0.04748286
   0.04670019 0.04835739 0.04689188]
  [0.04927909 0.04635496 0.04616063 0.04729576 0.04451634 0.04985309
   0.04795114 0.04636601 0.04961807 0.04548875 0.049214   0.04691025
   0.04623386 0.04797848 0.04819691 0.04892079 0.04958323 0.04519735
   0.04875692 0.0499779  0.04614647]
  [0.04757084 0.04526785 0.0442223  0.04853389 0.04532959 0.04738509
   0.04752239 0.04945779 0.04571299 0.04768217 0.04972003 0.04684024
   0.04906619 0.04652106 0.05080967 0.04421029 0.04934925 0.04793061
   0.04640952 0.05262494 0.04783323]
  [0.04732653 0.04782978 0.04627385 0.05304922 0.04496691 0.04709951
   0.04778071 0.04510077 0.04289808 0.04736049 0.04611911 0.04937108
   0.04802981 0.0501514  0.04800784 0.05166512 0.0501681  0.04507516
   0.04866832 0.04709613 0.04596211]]] (1, 4, 21)
3
Probability of Sentence " student and my " is: 0.00010385079157109967
```

```
   0.04828878 0.04893285 0.04524771 0.0529214  0.05041301 0.04769088
   0.0491938  0.0438129  0.04734754 0.04283423 0.04499082 0.04948651
   0.04790879 0.0483557  0.0475654 ]
  [0.0470341  0.04750969 0.0479287  0.04933253 0.04998367 0.04435573
   0.04699506 0.04518249 0.04539933 0.04669947 0.04775457 0.0510104
   0.04830541 0.04825601 0.04731837 0.04846344 0.05004127 0.04515551
   0.04915455 0.0476405  0.04647917]
  [0.05211634 0.04592865 0.04386634 0.04669205 0.04827831 0.05144821
   0.04913236 0.04457719 0.05193174 0.04827379 0.05232117 0.04347389
   0.04423366 0.04670035 0.04811227 0.04757246 0.05030061 0.04661572
   0.04495738 0.04812713 0.04534047]]] (1, 5, 21)
4
```