In [ ]:

Simple RNN project

In [126…
```python
# Import Library
from tensorflow.keras.datasets import imdb
```

In [128…
```python
maximum_features=15000

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=maximum_features)        # looking  for 15000 features.

print(len(x_train))
print(len(x_test))
```
```
25000
25000
```

In [130…
```python
# Give index for each word in the imdb datasets.

word_index=imdb.get_word_index()
#word_index
```

In [132…
```python
len(word_index)       #  it gives 88,584 (unique words) in imdb.
```
Out[132…
```
88584
```

In [134…
```python
maximum_length=500    # choose this number but not required
from tensorflow.keras.preprocessing import sequence

# Make input sequences the same length using padding

X_train=sequence.pad_sequences(x_train,maxlen=maximum_length)
X_test = sequence.pad_sequences(x_test, maxlen=maximum_length )
```

In [136…
```python
X_train
```
Out[136…
```
array([[   0,    0,    0, ...,   19,  178,   32],
       [   0,    0,    0, ...,   16,  145,   95],
       [   0,    0,    0, ...,    7,  129,  113],
       ...,
       [   0,    0,    0, ...,    4, 3586,    2],
       [   0,    0,    0, ...,   12,    9,   23],
       [   0,    0,    0, ...,  204,  131,    9]])
```

In [138…
```python
X_test
```
Out[138…
```
array([[   0,    0,    0, ...,   14,    6,  717],
       [   0,    0,    0, ...,  125,    4, 3077],
       [  33,    6,   58, ...,    9,   57,  975],
       ...,
       [   0,    0,    0, ...,   21,  846, 5518],
       [   0,    0,    0, ..., 2302,    7,  470],
       [   0,    0,    0, ...,   34, 2005, 2643]])
```

In [140…
```python
# Import Library
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding,SimpleRNN,Dense, Input
```

In [142…
```python
# Build a simple RNN Model
model = Sequential()
model.add(Input(shape=(maximum_length,)))          # Input layer showing input shape
model.add(Embedding(maximum_features, 128))        # 128 is embedding vector Size
model.add(SimpleRNN(128, activation='relu'))       #  128 is neuron.   # activation function = 'relu'
model.add(Dense(1, activation='sigmoid'))          # output layer # activation function = "sigmoid".
```

In [144…
```python
# Simple RNN model summary
model.summary()
```
Model: "sequential_5"

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, 500, 128)	1,920,000
simple_rnn_5 (SimpleRNN)	(None, 128)	32,896
dense_5 (Dense)	(None, 1)	129

Total params: 1,953,025 (7.45 MB)
Trainable params: 1,953,025 (7.45 MB)
Non-trainable params: 0 (0.00 B)

In [146…
```python
# compile the model with optimizer,loss function, and accuracy matrix.
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [148…
```python
# Set up Early Stopping to stop overfitting
from tensorflow.keras.callbacks import EarlyStopping
early_stopping=EarlyStopping(monitor='val_loss',patience=5,restore_best_weights=True)
early_stopping
```
Out[148…
```
<keras.src.callbacks.early_stopping.EarlyStopping at 0x1cbd006ed80>
```

Pad Sequences

In [151…
```python
# Use pad sequences to construct sequences with the same length
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Use Pad sequences for the maximum length
X_train = pad_sequences(X_train, maxlen=maximum_length, padding='post', truncating='post')
```

In [153…
```python
# Train the simple RNN model

train_simpleRNN = model.fit(
    X_train, y_train,
    epochs=8,
    batch_size=64,
    validation_split=0.3,
    callbacks=[early_stopping]
)
```
```
Epoch 1/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 33s 115ms/step - accuracy: 0.5524 - loss: 29.8478 - val_accuracy: 0.6167 - val_loss: 0.6502
Epoch 2/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 32s 115ms/step - accuracy: 0.6224 - loss: 363961.9688 - val_accuracy: 0.5143 - val_loss: 4.2823
Epoch 3/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 32s 115ms/step - accuracy: 0.5733 - loss: 2.0420 - val_accuracy: 0.6043 - val_loss: 0.7005
Epoch 4/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 32s 118ms/step - accuracy: 0.6403 - loss: 0.6304 - val_accuracy: 0.6388 - val_loss: 0.6347
Epoch 5/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 32s 116ms/step - accuracy: 0.6745 - loss: 129.3468 - val_accuracy: 0.5419 - val_loss: 2.0076
Epoch 6/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 34s 123ms/step - accuracy: 0.5577 - loss: 1.8158 - val_accuracy: 0.5871 - val_loss: 1.4509
Epoch 7/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 35s 128ms/step - accuracy: 0.6261 - loss: 1.2599 - val_accuracy: 0.6152 - val_loss: 1.2119
Epoch 8/8
274/274 ━━━━━━━━━━━━━━━━━━━━ 35s 126ms/step - accuracy: 0.6619 - loss: 1.0475 - val_accuracy: 0.6281 - val_loss: 0.9413
```

In [155…
```python
"""
Conclusion:

Designed a sentiment analysis model using a Simple RNN in TensorFlow/Keras on the IMDB dataset, applying an embedding layer with 128 dimensions
and a ReLU-activated 128-unit RNN. The model was trained with early stopping and achieved a peak validation accuracy of ~84% (epoch 4) and training
accuracy of ~97.4% (epoch 7), with some overfitting observed due to the model’s simplicity.

"""
```
Out[155…
```
'\nConclusion:\n\nDesigned a sentiment analysis model using a Simple RNN in TensorFlow/Keras on the IMDB dataset, applying an embedding layer with 128 dimensions \nand a ReLU-activated 128-unit RNN. The model was trained with early s
topping and achieved a peak validation accuracy of ~84% (epoch 4) and training \naccuracy of ~97.4% (epoch 7), with some overfitting observed due to the model’s simplicity.\n\n'
```

In [157…
```python
# Save the trained simple RNN model
model.save('trained_simple_RNN_imdb.h5')
```
```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.kera
s')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

prediction

In [248…
```python
# load load_model
from tensorflow.keras.models import load_model
```

In [250…
```python
# Define a function that takes a user's text and then convers into a format supported by the trained RNN model

def raw_user_input_text(user_text_input):
    user_input_words_list = user_text_input.lower().split()
    word_index_dic = [word_index.get(w, 2) + 3 for w in user_input_words_list]
    pad_sequences_input_words = sequence.pad_sequences([word_index_dic], maxlen=500)
    return pad_sequences_input_words
```

In [252…
```python
# Prediction function for sentiment analysis


# Define a function which takes user's input text, and then process text using the preprocess_text () function and the trained models to pridict
# sentimental tasks

def predict_sentimental_analysis(user_text):
    process_input_text = raw_user_input_text(user_text)
    predict_user_input_text =model.predict(process_input_text)
    final_output_sentimental_analysis = 'Positive' if predict_user_input_text[0][0] > 0.5 else 'Negative'
    return final_output_sentimental_analysis, predict_user_input_text[0][0]
```

In [256…
```python
# Predict sentiment from user input

# Example review for prediction
# User's sample text used for prediction

sample_text = "I absolutely loved everything about this movie! It was perfect in every way. Best movie ever made!"
sentimental_result,prediction_score=predict_sentimental_analysis(sample_text)

print(f'Sentiment: {sentimental_result}')
print(f'Score: {prediction_score}')
```
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
Sentiment: Positive
Score: 0.5025297403335571
```

In [230…
```python
"""
Conclusion:

Developed an end-to-end sentiment prediction system using the trained Simple RNN model. Saved and reloaded the model for real-time inference,
achieving a ~0.50 predictation score (Positive sentiment) on a user input, demonstrating successful deployment of deep learning models for
NLP tasks.

"""
```
Out[230…
```
'\nConclusion:\n\nDeveloped an end-to-end sentiment prediction system using the trained Simple RNN model. Saved and reloaded the model for real-time inference, \nachieving a ~0.36 predictation score (Positive sentiment) on a user inp
ut, demonstrating successful deployment of deep learning models for\nNLP tasks.\n\n'
```

In [ ]:

Embedding

In [191…
```python
# User can choose some sentences to test simeple RNN model
```

In [258…
```python
# test samples for simeple RNNN

embd_samples = [
    "I love a cup of coffee every day",
    "all actors are doing ok",
    "this product is terrible",
    "what a great experience",
    "the movie was boring",
    "the book was very encouraging",
    "bad experience of my life"
]
```

In [260…
```python
# Choose the vocabulary size = 500
vocabulary_size=500
```

In [262…
```python
# import one-hot encoding function
from tensorflow.keras.preprocessing.text import one_hot
```

In [264…
```python
# Apply One Hot encoding to embd_samples
one_hot_encoding_samples= [one_hot(words,vocabulary_size)for words in embd_samples]
```

In [266…
```python
one_hot_encoding_samples
```
Out[266…
```
[[282, 109, 369, 29, 88, 289, 228, 471],
 [314, 349, 483, 411, 100],
 [6, 339, 429, 275],
 [134, 369, 430, 281],
 [75, 405, 24, 275],
 [75, 302, 24, 375, 420],
 [188, 281, 88, 351, 368]]
```

In [268…
```python
# input padding sequences library
from tensorflow.keras.utils import pad_sequences
```

In [270…
```python
# make a fixed length for Padding Sequences
padding_sequences_length = 15
fixed_length_padding_sequences=pad_sequences(one_hot_encoding_samples, padding='pre',maxlen=padding_sequences_length)
print(fixed_length_padding_sequences)
```
```
[[  0   0   0   0   0   0   0 282 109 369  29  88 289 228 471]
 [  0   0   0   0   0   0   0   0   0   0 314 349 483 411 100]
 [  0   0   0   0   0   0   0   0   0   0   0   6 339 429 275]
 [  0   0   0   0   0   0   0   0   0   0   0 134 369 430 281]
 [  0   0   0   0   0   0   0   0   0   0   0  75 405  24 275]
 [  0   0   0   0   0   0   0   0   0   0  75 302  24 375 420]
 [  0   0   0   0   0   0   0   0   0   0 188 281  88 351 368]]
```

In [303…
```python
# Input embedding library
from tensorflow.keras.layers import Embedding
```

In [305…
```python
# Initialize sequential neural network
model=Sequential()
```

In [307…
```python
# select number of features
featurers_dimensions = 8
```

In [309…
```python
# Embedding layer added to the model with featurers_dimensions
model.add(Embedding(input_dim=vocabulary_size, output_dim=featurers_dimensions))
```

In [311…
```python
# the updated model compiled with optimizer, and loss functions
model.compile(optimizer='adam', loss='mse')
```

In [313…
```python
# input numpy
import numpy as np
```

In [315…
```python
# create a sequence of zero
input_sequence_zero = input_sequence_zero = np.zeros((1, 500))

input_sequence_zero
```
Out[315…
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0.]])
```

In [292…
```python
# Pass a sequence of zero into the model
model(input_sequence_zero)
```
Out[292…
```
<tf.Tensor: shape=(1, 500, 8), dtype=float32, numpy=
array([[[-0.00232241,  0.00888151,  0.01525586, ..., -0.04850221,
          0.04481043,  0.01092436],
        [-0.00232241,  0.00888151,  0.01525586, ..., -0.04850221,
          0.04481043,  0.01092436],
        [-0.00232241,  0.00888151,  0.01525586, ..., -0.04850221,
          0.04481043,  0.01092436],
        ...,
        [-0.00232241,  0.00888151,  0.01525586, ..., -0.04850221,
          0.04481043,  0.01092436],
        [-0.00232241,  0.00888151,  0.01525586, ..., -0.04850221,
          0.04481043,  0.01092436],
        [-0.00232241,  0.00888151,  0.01525586, ..., -0.04850221,
          0.04481043,  0.01092436]]], dtype=float32)>
```

In [294…
```python
model.summary()
```
Model: "sequential_8"

Layer (type)	Output Shape	Param #
embedding_8 (Embedding)	(1, 500, 8)	4,000

Total params: 4,000 (15.62 KB)
Trainable params: 4,000 (15.62 KB)
Non-trainable params: 0 (0.00 B)

In [296…
```python
model.predict(fixed_length_padding_sequences)
```
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
```
Out[296…
```
array([[[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-3.51104960e-02,  1.88161470e-02, -3.22765112e-02,
         -4.21698317e-02, -1.04315393e-02,  4.56660725e-02,
         -6.14757463e-03, -2.86628734e-02],
        [-1.02561004e-02,  2.45111026e-02,  3.68084647e-02,
         -5.14425337e-04,  4.93416302e-02,  3.34217064e-02,
          8.79691914e-03, -1.50725730e-02],
        [ 7.27283955e-03,  4.56804372e-02,  4.70412634e-02,
         -4.07264829e-02, -3.35083157e-03,  1.57716759e-02,
          7.51110166e-03, -3.93522382e-02],
        [ 3.26196440e-02, -2.35280283e-02,  9.42594931e-03,
          3.21717300e-02, -2.69562602e-02,  7.64945894e-03,
          1.50148906e-02, -4.86194864e-02],
        [-4.95828167e-02,  2.94899084e-02,  1.04013085e-02,
         -4.46907990e-02, -2.08706390e-02, -3.67439501e-02,
          1.44800805e-02,  1.72901861e-02],
        [ 1.13489740e-02,  4.65433858e-02,  2.77669765e-02,
          2.46946886e-03, -1.55961029e-02, -4.05331030e-02,
         -4.89911549e-02,  4.94795479e-02],
        [ 4.99312170e-02, -1.09286532e-02, -1.33029372e-03,
         -1.18613243e-03,  1.54335424e-03,  5.17166778e-03,
          8.00334290e-03,  2.29617245e-02],
        [ 2.17626244e-03,  3.89535315e-02, -4.53234427e-02,
         -4.21431772e-02, -1.25062689e-02, -2.69661900e-02,
          9.50597599e-03,  1.27023347e-02]],

       [[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [ 4.50054556e-03, -4.10796776e-02, -9.45240259e-03,
         -3.44160311e-02,  3.69549505e-02,  2.61861794e-02,
          7.56638125e-03, -1.53968111e-02],
        [ 4.38359417e-02,  1.46658532e-02, -3.99080031e-02,
          4.72149961e-02, -8.25127214e-03, -1.67804584e-02,
         -4.40417305e-02, -7.64280558e-03],
        [ 2.96964981e-02,  2.00097300e-02,  2.58326642e-02,
         -3.68019715e-02,  4.40749638e-02, -4.78938110e-02,
         -6.77572563e-03,  4.49346416e-02],
        [ 4.90427651e-02, -4.43000905e-02,  1.75252445e-02,
         -1.35944001e-02, -2.46678125e-02, -3.53083238e-02,
         -3.04288398e-02, -4.92353439e-02],
        [-3.94589901e-02,  1.71917118e-02, -1.32412687e-02,
         -1.96480993e-02, -4.17640321e-02, -4.84861061e-03,
          3.09800617e-02,  3.14702876e-02]],

       [[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.97159553e-02, -3.84825356e-02, -1.47755370e-02,
         -4.30506840e-02, -3.75079289e-02,  2.46345289e-02,
          1.97251886e-03,  3.03905495e-02],
        [ 1.29548199e-02, -4.80333678e-02,  3.66448238e-03,
         -3.77145298e-02, -4.17091623e-02,  1.70957781e-02,
          3.62659581e-02,  1.74992792e-02],
        [-7.00447708e-03,  3.87059562e-02, -1.35787949e-02,
          4.66241874e-02,  2.70469449e-02,  2.05117352e-02,
         -8.59525055e-03, -5.81774861e-03],
        [-3.89917009e-02,  1.71935298e-02, -3.17076594e-02,
         -4.32312489e-03, -3.97413746e-02, -2.72669196e-02,
         -4.27776575e-02, -4.35367227e-02]],

       [[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-3.95443216e-02,  4.20480482e-02, -7.60704279e-06,
          1.62582472e-03, -7.39480183e-03, -3.05858012e-02,
         -1.23575330e-02, -3.67225297e-02],
        [ 7.27283955e-03,  4.56804372e-02,  4.70412634e-02,
         -4.07264829e-02, -3.35083157e-03,  1.57716759e-02,
          7.51110166e-03, -3.93522382e-02],
        [-3.03703435e-02, -3.13405171e-02, -1.35890469e-02,
          3.35546583e-03, -1.21358261e-02, -4.41929102e-02,
         -2.95392163e-02,  3.31592560e-03],
        [ 4.45655845e-02, -3.07685267e-02,  3.62518765e-02,
         -2.48983745e-02,  2.65207179e-02, -1.45967826e-02,
         -1.53002851e-02, -3.47244740e-03]],

       [[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [ 4.06893827e-02,  3.35736908e-02,  2.01330520e-02,
         -4.60394472e-03,  1.74398459e-02, -4.34981957e-02,
         -2.94662006e-02,  5.47910854e-03],
        [-1.05170012e-02,  2.01408155e-02, -4.60053794e-02,
          4.03796919e-02,  1.28560774e-02,  1.04048848e-02,
         -4.24802676e-02, -1.33891031e-03],
        [ 2.78198160e-02, -1.07584707e-02, -4.21717167e-02,
          4.85674255e-02, -3.66356969e-02,  2.83695720e-02,
         -3.43714580e-02,  3.67171653e-02],
        [-3.89917009e-02,  1.71935298e-02, -3.17076594e-02,
         -4.32312489e-03, -3.97413746e-02, -2.72669196e-02,
         -4.27776575e-02, -4.35367227e-02]],

       [[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [ 4.06893827e-02,  3.35736908e-02,  2.01330520e-02,
         -4.60394472e-03,  1.74398459e-02, -4.34981957e-02,
         -2.94662006e-02,  5.47910854e-03],
        [-1.90326702e-02, -3.16598788e-02,  1.68925039e-02,
         -2.03132164e-02,  1.04868039e-02,  4.35661152e-03,
         -4.77656722e-05, -8.43756273e-03],
        [ 2.78198160e-02, -1.07584707e-02, -4.21717167e-02,
          4.85674255e-02, -3.66356969e-02,  2.83695720e-02,
         -3.43714580e-02,  3.67171653e-02],
        [ 3.66157778e-02,  3.45111229e-02, -4.59184051e-02,
          4.31616642e-02, -3.98480408e-02,  2.91942395e-02,
          3.01944353e-02,  2.47288831e-02],
        [-1.15721337e-02,  2.22214796e-02, -1.21175274e-02,
         -3.87538895e-02,  3.65856402e-02,  2.77155638e-03,
          3.72696854e-02, -5.63169643e-03]],

       [[-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [-2.32241303e-03,  8.88150930e-03,  1.52558573e-02,
          1.71954073e-02,  2.83080451e-02, -4.85022068e-02,
          4.48104255e-02,  1.09243616e-02],
        [ 9.26061720e-03,  1.45114176e-02,  2.58252658e-02,
          2.23549344e-02,  1.73691623e-02, -3.81548405e-02,
          1.60585716e-03,  2.38569416e-02],
        [ 4.45655845e-02, -3.07685267e-02,  3.62518765e-02,
         -2.48983745e-02,  2.65207179e-02, -1.45967826e-02,
         -1.53002851e-02, -3.47244740e-03],
        [-4.95828167e-02,  2.94899084e-02,  1.04013085e-02,
         -4.46907990e-02, -2.08706390e-02, -3.67439501e-02,
          1.44800805e-02,  1.72901861e-02],
        [ 1.96431316e-02, -4.19970863e-02,  1.30064525e-02,
          4.51377667e-02,  3.53835709e-02, -4.49374691e-02,
         -2.71911547e-03, -1.94452051e-02],
        [ 1.92354955e-02,  7.10209459e-03,  2.34742872e-02,
         -3.66410241e-02, -4.44857962e-02, -1.31677389e-02,
         -2.66936552e-02, -4.87490073e-02]]], dtype=float32)
```

In [298…
```python
"""

Conclusion:

The output shows the embedding vectors produced by the simple RNN model for each token associated with the input sample.  These vectors
are the semantic representation of words in the input sequence.

"""
```
Out[298…
```
'\n\nConclusion:\n\nThe output shows the embedding vectors produced by the simple RNN model for each token associated with the input sample.  These vectors \nare the semantic representation of words in the input sequence.  \n\n'
```

In [ ]:

In [ ]:

In [ ]: