HW4

Yasaman Emami Deep Learning Dec2,2022

1 Preprocessing is done using:
   a. Tokenization: Spacy package tokenizer
   b. Lower case: lower() function as a python built-in function is applied
   c. Contraction: contractions.fix() is applid from contractions package
   d. Remove punctuations: filter based method is applied if a token is not in '\n\n \n\n\n!"-#$%&()--.*+,-/:;<=>?@[\\]^_`{|}~\t\n ' string then pass through the filter
   e. Stemming and lemmatizing: PorterStemmer() & WordNetLemmatizer() from nltk package

2 Text Generation using Keras word embedding layer:

An embedding layer with vocab_size which is equal to tokenizer.word_counts with embedding size of 25

The second layer is lstm with 150 units as dimensionality of the output space and return_sequences set to True which means it doesn't only return the output of last layer.

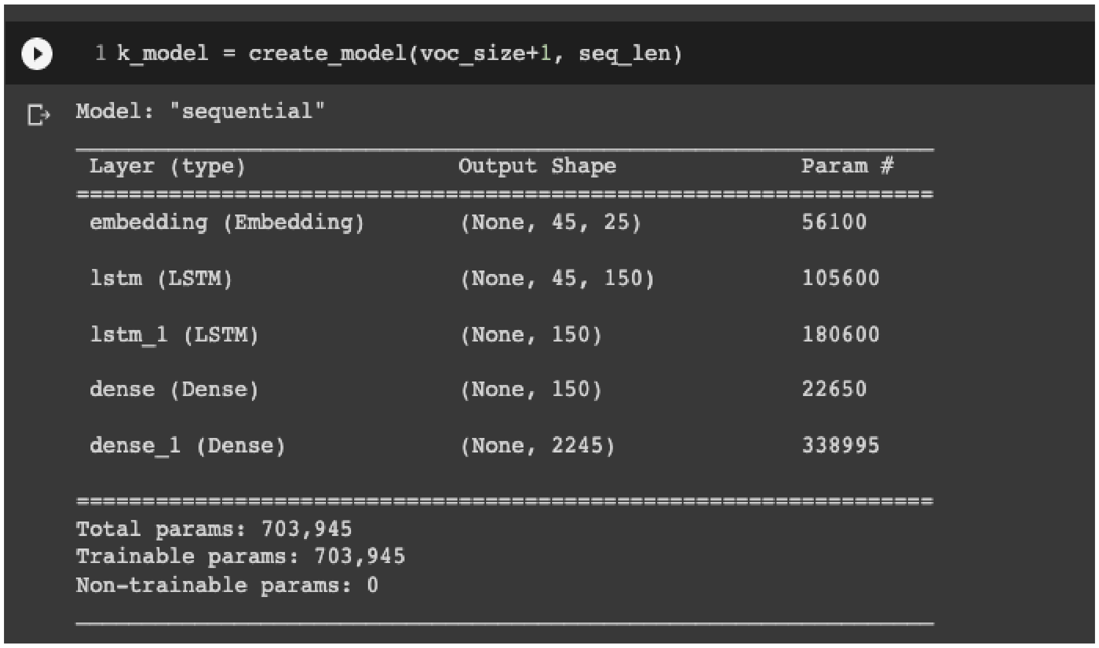
Third layer is another lstm with 150 untis.

Fourth layer is a dense layer with 150 units as output and 'relu' as activation.

The last layer is a dense with output as much as voicabulary size and activation as softmax.

The loss is categorical cross entropy because the labels are one-hot encoded, the optimizer is 'adam' for this model.

Below is the summary of the model:

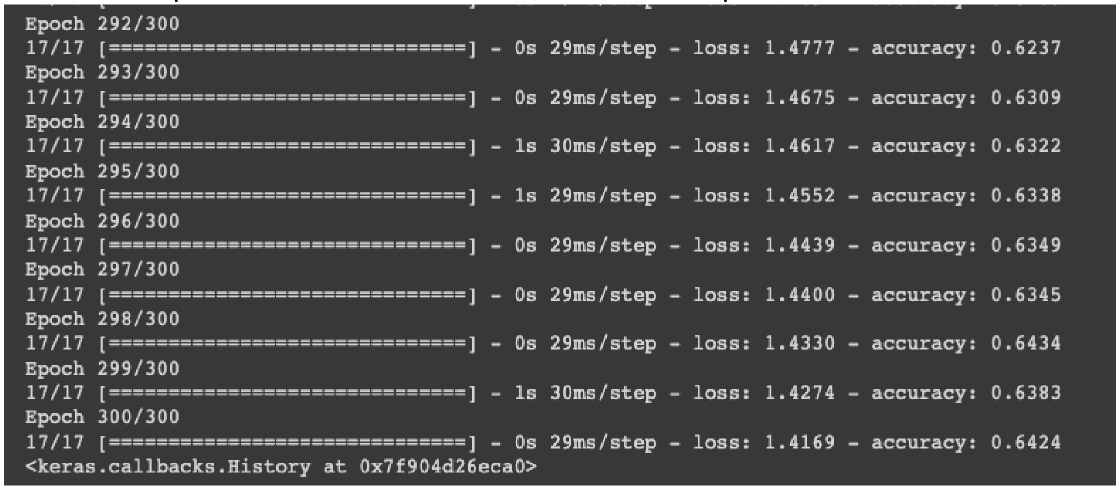


```
k_model = create_model(voc_size+1, seq_len)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 45, 25)            56100

 lstm (LSTM)                 (None, 45, 150)           105600

 lstm_1 (LSTM)               (None, 150)               180600

 dense (Dense)               (None, 150)               22650

 dense_1 (Dense)             (None, 2245)              338995

=================================================================
Total params: 703,945
Trainable params: 703,945
Non-trainable params: 0
_________________________________________________________________
```

Fig1. **Keras** Embedding Model text generation

then train the model with model.fit() X which is coming from sequences of 45 words and the 46 word as y is feed to fit method to train, batch_size is selected as 512 and run for 300 epochs. Below are the results are few last epochs:



```
Epoch 292/300
17/17 [==============================] - 0s 29ms/step - loss: 1.4777 - accuracy: 0.6237
Epoch 293/300
17/17 [==============================] - 0s 29ms/step - loss: 1.4675 - accuracy: 0.6309
Epoch 294/300
17/17 [==============================] - 1s 30ms/step - loss: 1.4617 - accuracy: 0.6322
Epoch 295/300
17/17 [==============================] - 1s 29ms/step - loss: 1.4552 - accuracy: 0.6338
Epoch 296/300
17/17 [==============================] - 0s 29ms/step - loss: 1.4439 - accuracy: 0.6349
Epoch 297/300
17/17 [==============================] - 0s 29ms/step - loss: 1.4400 - accuracy: 0.6345
Epoch 298/300
17/17 [==============================] - 0s 29ms/step - loss: 1.4330 - accuracy: 0.6434
Epoch 299/300
17/17 [==============================] - 1s 30ms/step - loss: 1.4274 - accuracy: 0.6383
Epoch 300/300
17/17 [==============================] - 0s 29ms/step - loss: 1.4169 - accuracy: 0.6424
<keras.callbacks.History at 0x7f904d26eca0>
```

Fig2. Few Epochs of **Keras** Embedding Model text generation

3 Text Generation with Glove Embedding.

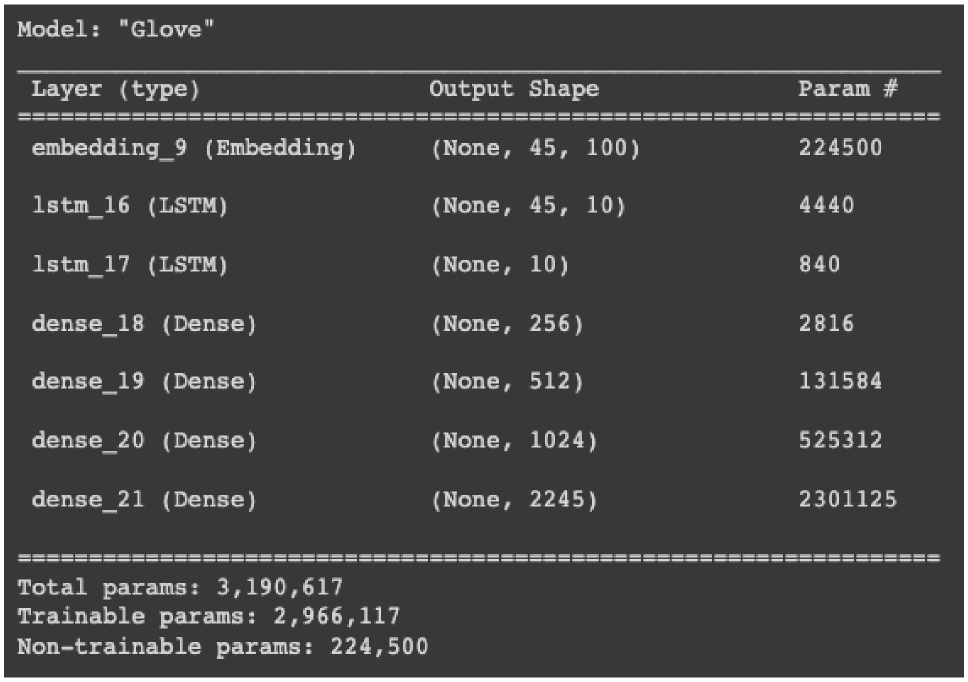
Pretrained 100-dimension model is applied to get the vectors of mobydick data. The embedding matrix from embedding vectors is then constructed and later fed to embedding layer of the model as weights.

The sequential model for this approach is generated starting with embedding layer with embedding size of 100 weights from glove embedding matrix and 'trainable' is set to False to freeze the layer so that this layer wouldn't be updated druring the training process. Then few lstm and dense layers are added to the model and the last layer is the size of vocab with 'softmax' activation.

The optimizer for this model is RMSprop with learning rate of 0.001, although adam works better and it did for the previous model using adam was showing very low accuracy which was not getting any better even after 30 epochs and RMSprop optimizer was applied due to the higher accuracy level.

The model then gets train with the same number of tokens as X(45) and the 46th token as 'y'. epochs set to 300 and batch is 256.

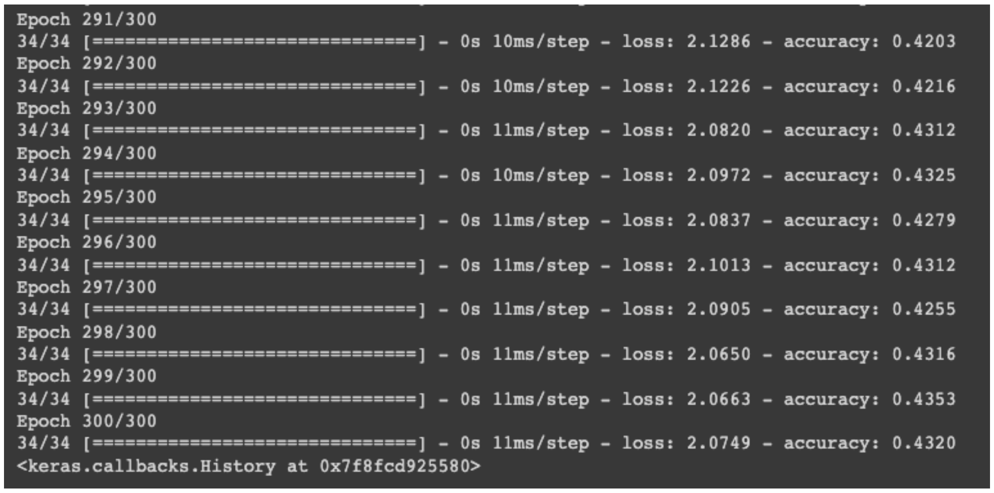
Summary of the model is as below:



```
Model: "Glove"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 45, 100)           224500

 lstm_16 (LSTM)              (None, 45, 10)            4440

 lstm_17 (LSTM)              (None, 10)                840

 dense_18 (Dense)            (None, 256)               2816

 dense_19 (Dense)            (None, 512)               131584

 dense_20 (Dense)            (None, 1024)              525312

 dense_21 (Dense)            (None, 2245)              2301125

=================================================================
Total params: 3,190,617
Trainable params: 2,966,117
Non-trainable params: 224,500
```

Fig3. **Glove** Embedding Model text generation

and the few last epochs out of 300 results are as below:



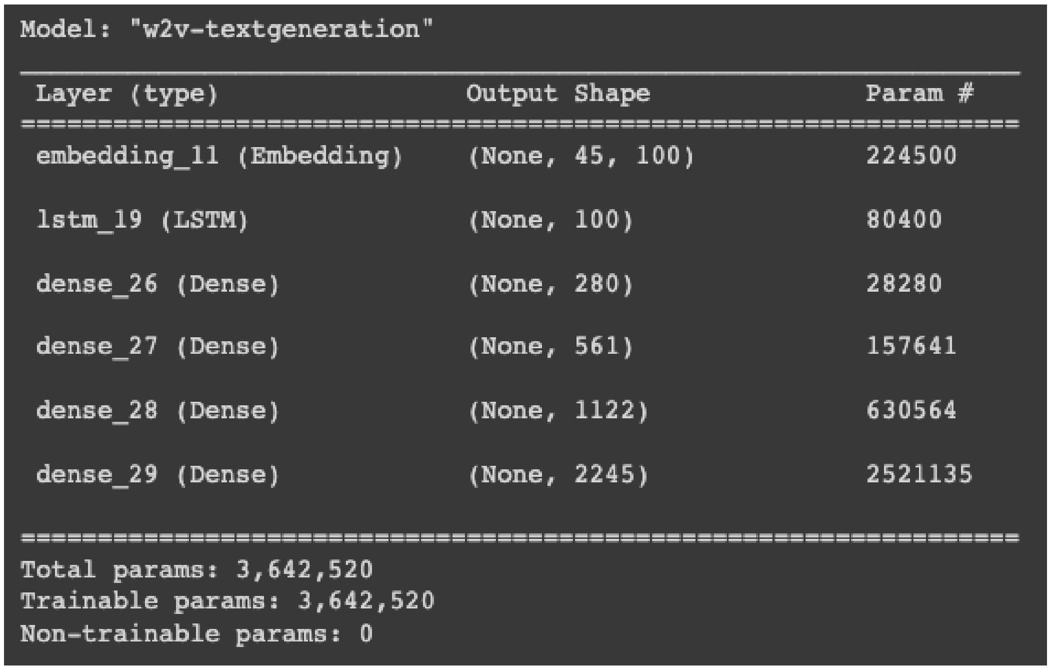
```
Epoch 291/300
34/34 [==============================] - 0s 10ms/step - loss: 2.1286 - accuracy: 0.4203
Epoch 292/300
34/34 [==============================] - 0s 10ms/step - loss: 2.1226 - accuracy: 0.4216
Epoch 293/300
34/34 [==============================] - 0s 11ms/step - loss: 2.0820 - accuracy: 0.4312
Epoch 294/300
34/34 [==============================] - 0s 10ms/step - loss: 2.0972 - accuracy: 0.4325
Epoch 295/300
34/34 [==============================] - 0s 11ms/step - loss: 2.0837 - accuracy: 0.4279
Epoch 296/300
34/34 [==============================] - 0s 11ms/step - loss: 2.1013 - accuracy: 0.4312
Epoch 297/300
34/34 [==============================] - 0s 11ms/step - loss: 2.0905 - accuracy: 0.4255
Epoch 298/300
34/34 [==============================] - 0s 11ms/step - loss: 2.0650 - accuracy: 0.4316
Epoch 299/300
34/34 [==============================] - 0s 11ms/step - loss: 2.0663 - accuracy: 0.4353
Epoch 300/300
34/34 [==============================] - 0s 11ms/step - loss: 2.0749 - accuracy: 0.4320
<keras.callbacks.History at 0x7f8fcd925580>
```

Fig4. Few epochs of **Glove** Embedding Model text generation

4 Text Generation with wor2vec as Embedding:

And embedding matrix with weights from wv model is constructed the embedding size is set to 100 and a sequential model based on the w2v as embedding layer is built.
The input size is the vocabulary size, with the oupt put of embedding size and the weights coming from embedding matrix also the input len is set to 45.

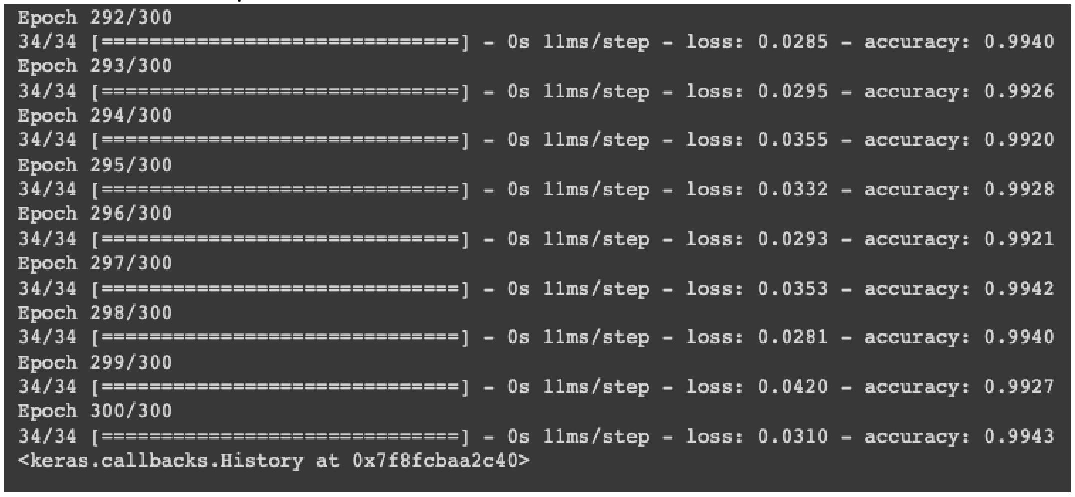
The few lstm and dense layers are added with the last layer same as previous models of vocab_size output and 'softmax' as activation the optimizer is RMSprop and loss is categorical_cross entropy
Then the model gets train with the same X, y as previous models and epochs of 300 and batch size of 256.
The w2v model is as below:



```
Model: "w2v-textgeneration"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 45, 100)           224500

 lstm_19 (LSTM)              (None, 100)               80400

 dense_26 (Dense)            (None, 280)               28280

 dense_27 (Dense)            (None, 561)               157641

 dense_28 (Dense)            (None, 1122)              630564

 dense_29 (Dense)            (None, 2245)              2521135

=================================================================
Total params: 3,642,520
Trainable params: 3,642,520
Non-trainable params: 0
```

Fig5. **Word2Vec** Embedding Model text generation

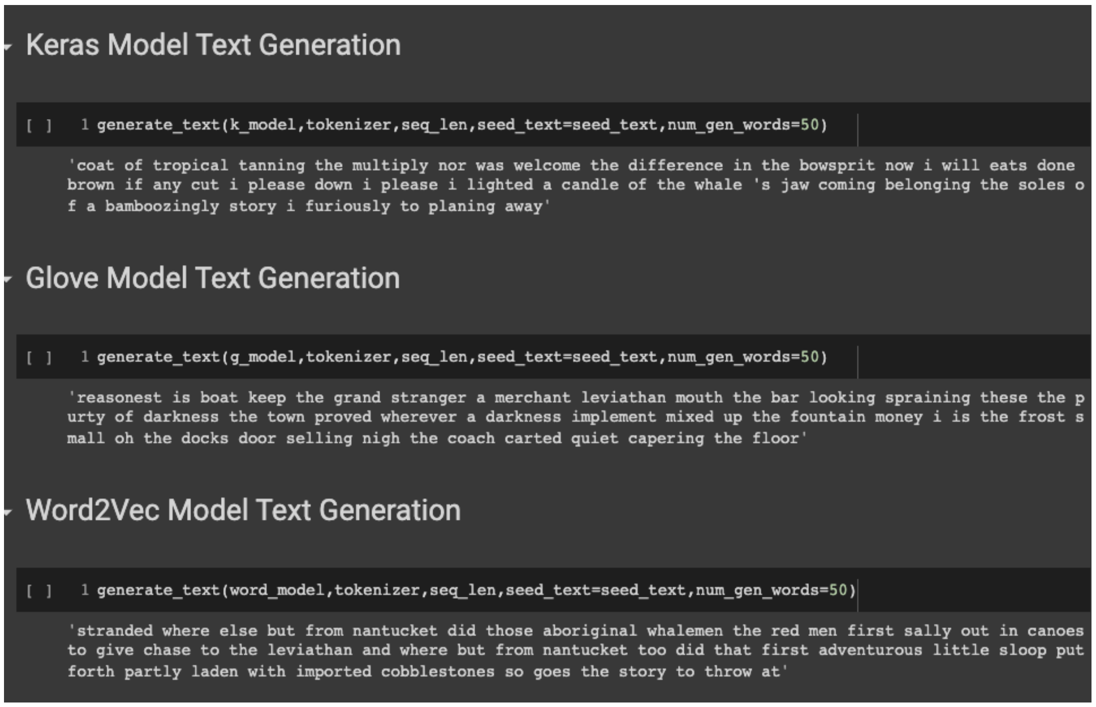
and the few last epochs are as below:



```
Epoch 292/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0285 - accuracy: 0.9940
Epoch 293/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0295 - accuracy: 0.9926
Epoch 294/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0355 - accuracy: 0.9920
Epoch 295/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0332 - accuracy: 0.9928
Epoch 296/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0293 - accuracy: 0.9921
Epoch 297/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0353 - accuracy: 0.9942
Epoch 298/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0281 - accuracy: 0.9940
Epoch 299/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0420 - accuracy: 0.9927
Epoch 300/300
34/34 [==============================] - 0s 11ms/step - loss: 0.0310 - accuracy: 0.9943
<keras.callbacks.History at 0x7f8fcbaa2c40>
```

Fig6. Few epochs of **Word2Vec** Embedding Model text generation

the results from generating text from the models with 50 words as number of generated words are as below:



Keras Model Text Generation

```
generate_text(k_model,tokenizer,seq_len,seed_text=seed_text,num_gen_words=50)
```

```
'coat of tropical tanning the multiply nor was welcome the difference in the bowsprit now i will eats done
brown if any cut i please down i please i lighted a candle of the whale 's jaw coming belonging the soles o
f a bamboozingly story i furiously to planing away'
```

Glove Model Text Generation

```
generate_text(g_model,tokenizer,seq_len,seed_text=seed_text,num_gen_words=50)
```

```
'reasonest is boat keep the grand stranger a merchant leviathan mouth the bar looking spraining these the p
urty of darkness the town proved wherever a darkness implement mixed up the fountain money i is the frost s
mall oh the docks door selling nigh the coach carted quiet capering the floor'
```

Word2Vec Model Text Generation

```
generate_text(word_model,tokenizer,seq_len,seed_text=seed_text,num_gen_words=50)
```

```
'stranded where else but from nantucket did those aboriginal whalemen the red men first sally out in canoes
to give chase to the leviathan and where but from nantucket too did that first adventurous little sloop put
forth partly laden with imported cobblestones so goes the story to throw at'
```

Fig7. Generated text by different models