BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Group No: 106

Full Name	BITS ID
Varshini	2021FC04149
Abhishek Shah	2021FC04153
Ritresh Girdhar	2021FC04145

Question and Answer Chat Bots

Loading the Data

We will be working with the Babi Data Set from Facebook Research.

Full Details: https://research.fb.com/downloads/babi/

- Jason Weston, Antoine Bordes, Sumit Chopra, Tomas Mikolov, Alexander M. Rush, "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks", http://arxiv.org/abs/1502.05698

In [7]:

```
import pickle
import numpy as np
```

Q1. Write code to unpickle the train_qa and test_qa datasets below

In [8]:

```
!pip install keras
!pip install tensorflow
!pip install Keras-Preprocessing
!pip install embeddings
!pip install keras-pos-embd
```

```
Requirement already satisfied: keras in /Users/ritgirdh/opt/anaconda3/lib/p
ython3.9/site-packages (2.11.0)
Collecting tensorflow
  Downloading tensorflow-2.11.1-cp39-cp39-macosx_10_14_x86_64.whl (244.3 M
B)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244.3/244.3 MB 5.3 MB/s eta 0:
00:0000:0100:01
Requirement already satisfied: six>=1.12.0 in /Users/ritgirdh/opt/anaconda
3/lib/python3.9/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: keras<2.12,>=2.11.0 in /Users/ritgirdh/opt/a
naconda3/lib/python3.9/site-packages (from tensorflow) (2.11.0)
Requirement already satisfied: tensorboard<2.12,>=2.11 in /Users/ritgirdh/o
pt/anaconda3/lib/python3.9/site-packages (from tensorflow) (2.11.2)
Requirement already satisfied: libclang>=13.0.0 in /Users/ritgirdh/opt/anac
onda3/lib/python3.9/site-packages (from tensorflow) (15.0.6.1)
Requirement already satisfied: h5py>=2.9.0 in /Users/ritgirdh/opt/anaconda
3/lib/python3.9/site-packages (from tensorflow) (3.8.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /Users/ritgirdh/op
t/anaconda3/lib/python3.9/site-packages (from tensorflow) (3.19.6)
Requirement already satisfied: termcolor>=1.1.0 in /Users/ritgirdh/opt/anac
onda3/lib/python3.9/site-packages (from tensorflow) (2.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /Users/ritgirdh/opt/a
naconda3/lib/python3.9/site-packages (from tensorflow) (1.51.3)
Requirement already satisfied: packaging in /Users/ritgirdh/opt/anaconda3/l
ib/python3.9/site-packages (from tensorflow) (23.0)
Requirement already satisfied: setuptools in /Users/ritgirdh/opt/anaconda3/
lib/python3.9/site-packages (from tensorflow) (67.6.0)
Requirement already satisfied: astunparse>=1.6.0 in /Users/ritgirdh/opt/ana
conda3/lib/python3.9/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: wrapt>=1.11.0 in /Users/ritgirdh/opt/anacond
a3/lib/python3.9/site-packages (from tensorflow) (1.15.0)
Requirement already satisfied: flatbuffers>=2.0 in /Users/ritgirdh/opt/anac
onda3/lib/python3.9/site-packages (from tensorflow) (23.3.3)
Requirement already satisfied: typing-extensions>=3.6.6 in /Users/ritgirdh/
opt/anaconda3/lib/python3.9/site-packages (from tensorflow) (4.5.0)
Requirement already satisfied: google-pasta>=0.1.1 in /Users/ritgirdh/opt/a
naconda3/lib/python3.9/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /Use
rs/ritgirdh/opt/anaconda3/lib/python3.9/site-packages (from tensorflow) (0.
31.0)
Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in /User
s/ritgirdh/opt/anaconda3/lib/python3.9/site-packages (from tensorflow) (2.1
1.0)
Requirement already satisfied: numpy>=1.20 in /Users/ritgirdh/opt/anaconda
3/lib/python3.9/site-packages (from tensorflow) (1.24.2)
Requirement already satisfied: absl-py>=1.0.0 in /Users/ritgirdh/opt/anacon
da3/lib/python3.9/site-packages (from tensorflow) (1.4.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /Users/ritgirdh/opt/a
naconda3/lib/python3.9/site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /Users/ritgirdh/opt/ana
conda3/lib/python3.9/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /Users/ritgirdh/opt/an
aconda3/lib/python3.9/site-packages (from astunparse>=1.6.0->tensorflow)
(0.40.0)
Requirement already satisfied: markdown>=2.6.8 in /Users/ritgirdh/opt/anaco
nda3/lib/python3.9/site-packages (from tensorboard<2.12,>=2.11->tensorflow)
```

```
(3.4.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /Us
ers/ritgirdh/opt/anaconda3/lib/python3.9/site-packages (from tensorboard<2.
12,>=2.11->tensorflow) (0.6.1)
Requirement already satisfied: werkzeug>=1.0.1 in /Users/ritgirdh/opt/anaco
nda3/lib/python3.9/site-packages (from tensorboard<2.12,>=2.11->tensorflow)
(2.2.3)
Requirement already satisfied: google-auth<3,>=1.6.3 in /Users/ritgirdh/op
t/anaconda3/lib/python3.9/site-packages (from tensorboard<2.12,>=2.11->tens
orflow) (2.16.2)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /Users/ritg
irdh/opt/anaconda3/lib/python3.9/site-packages (from tensorboard<2.12,>=2.1
1->tensorflow) (1.8.1)
Requirement already satisfied: requests<3,>=2.21.0 in /Users/ritgirdh/opt/a
naconda3/lib/python3.9/site-packages (from tensorboard<2.12,>=2.11->tensorf
low) (2.28.2)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /Users/r
itgirdh/opt/anaconda3/lib/python3.9/site-packages (from tensorboard<2.12,>=
2.11->tensorflow) (0.4.6)
Requirement already satisfied: rsa<5,>=3.1.4 in /Users/ritgirdh/opt/anacond
a3/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensorboard<2.1
2,>=2.11->tensorflow) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /Users/ritgirdh/op
t/anaconda3/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensor
board<2.12,>=2.11->tensorflow) (5.3.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /Users/ritgirdh/op
t/anaconda3/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensor
board<2.12,>=2.11->tensorflow) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /Users/ritgirdh/
opt/anaconda3/lib/python3.9/site-packages (from google-auth-oauthlib<0.5,>=
0.4.1->tensorboard<2.12,>=2.11->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /Users/ritgirdh/o
pt/anaconda3/lib/python3.9/site-packages (from markdown>=2.6.8->tensorboard
<2.12,>=2.11->tensorflow) (6.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /Users/ritgirdh/opt/an
aconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard<
2.12,>=2.11->tensorflow) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/ritgirdh/op
t/anaconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorbo
ard<2.12,>=2.11->tensorflow) (1.26.15)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/ritgirdh/
opt/anaconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensor
board<2.12,>=2.11->tensorflow) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /Users/ritgirdh/opt/anaconda
3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>
=2.11->tensorflow) (3.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/ritgirdh/opt/ana
conda3/lib/python3.9/site-packages (from werkzeug>=1.0.1->tensorboard<2.12,
>=2.11->tensorflow) (2.1.2)
Requirement already satisfied: zipp>=0.5 in /Users/ritgirdh/opt/anaconda3/l
ib/python3.9/site-packages (from importlib-metadata>=4.4->markdown>=2.6.8->
tensorboard<2.12,>=2.11->tensorflow) (3.15.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /Users/ritgirdh/opt/
anaconda3/lib/python3.9/site-packages (from pyasn1-modules>=0.2.1->google-a
uth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /Users/ritgirdh/opt/anaco
```

```
nda3/lib/python3.9/site-packages (from requests-oauthlib>=0.7.0->google-aut
h-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow) (3.2.2)
Installing collected packages: tensorflow
Successfully installed tensorflow-2.11.1
Requirement already satisfied: Keras-Preprocessing in /Users/ritgirdh/opt/a
naconda3/lib/python3.9/site-packages (1.1.2)
Requirement already satisfied: numpy>=1.9.1 in /Users/ritgirdh/opt/anaconda
3/lib/python3.9/site-packages (from Keras-Preprocessing) (1.24.2)
Requirement already satisfied: six>=1.9.0 in /Users/ritgirdh/opt/anaconda3/
lib/python3.9/site-packages (from Keras-Preprocessing) (1.16.0)
Requirement already satisfied: embeddings in /Users/ritgirdh/opt/anaconda3/
lib/python3.9/site-packages (0.0.8)
Requirement already satisfied: tqdm in /Users/ritgirdh/opt/anaconda3/lib/py
thon3.9/site-packages (from embeddings) (4.62.3)
Requirement already satisfied: numpy in /Users/ritgirdh/opt/anaconda3/lib/p
ython3.9/site-packages (from embeddings) (1.24.2)
Requirement already satisfied: requests in /Users/ritgirdh/opt/anaconda3/li
b/python3.9/site-packages (from embeddings) (2.28.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/ritgirdh/
opt/anaconda3/lib/python3.9/site-packages (from requests->embeddings) (3.1.
0)
Requirement already satisfied: certifi>=2017.4.17 in /Users/ritgirdh/opt/an
aconda3/lib/python3.9/site-packages (from requests->embeddings) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/ritgirdh/op
t/anaconda3/lib/python3.9/site-packages (from requests->embeddings) (1.26.1
5)
Requirement already satisfied: idna<4,>=2.5 in /Users/ritgirdh/opt/anaconda
3/lib/python3.9/site-packages (from requests->embeddings) (3.4)
Requirement already satisfied: keras-pos-embd in /Users/ritgirdh/opt/anacon
da3/lib/python3.9/site-packages (0.13.0)
Requirement already satisfied: numpy in /Users/ritgirdh/opt/anaconda3/lib/p
ython3.9/site-packages (from keras-pos-embd) (1.24.2)
```

In [4]:

```
!pip install --upgrade --force-reinstall tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.11.1-cp39-cp39-macosx_10_14_x86_64.whl (244.3 M
B)
     ━━━━━━━━━━━━━━━━╸━━━━━━━━━━━━━━━━━━━━━━━ 100.3/244.3 MB 12.1 MB/s eta 0:
00:12ERROR: Could not install packages due to an OSError: [Errno 28] No spa
ce left on device

     ━━━━━━━━━━━━━━━━╸━━━━━━━━━━━━━━━━━━━━━━━ 100.3/244.3 MB 12.1 MB/s eta 0:
00:12
```

In [9]:

```
pip show tensorflow
```

```
Name: tensorflow
Version: 2.11.1
Summary: TensorFlow is an open source machine learning framework for everyo
ne.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
Location: /Users/ritgirdh/opt/anaconda3/lib/python3.9/site-packages
Requires: absl-py, astunparse, flatbuffers, gast, google-pasta, grpcio, h5p
y, keras, libclang, numpy, opt-einsum, packaging, protobuf, setuptools, si
x, tensorboard, tensorflow-estimator, tensorflow-io-gcs-filesystem, termcol
or, typing-extensions, wrapt
Required-by:
Note: you may need to restart the kernel to use updated packages.
```

In [10]:

```
# Keras library imports for tokenization and model building
from keras_preprocessing.sequence import pad_sequences
from keras_preprocessing.text import Tokenizer
from keras.models import Sequential, Model
from keras.layers import Embedding
from keras.layers import Input, Activation, Dense, Permute, Dropout
from keras.layers import add, dot, concatenate
from keras.layers import LSTM
from keras.layers import Embedding
```

```
2023-03-21 00:35:24.637006: I tensorflow/core/platform/cpu_feature_guard.c
c:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
/Users/ritgirdh/opt/anaconda3/lib/python3.9/site-packages/scipy/__init__.p
y:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for th
is version of SciPy (detected version 1.24.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [11]:

```
# Unpickling training and test dataset
with open('train_qa.txt', 'rb') as f:
    train_data = pickle.load(f)

with open('test_qa.txt', 'rb') as f:
    test_data = pickle.load(f)
```

Exploring the Format of the Data

Q2. Identify the type of traindata and test data

In [12]:

```
#type of train data
type(train_data)
```

Out[12]:

```
list
```

```
In [13]: #type of test data
         type(test_data)

Out[13]: list
```

Q3. Identify the length of test and train data

```
In [14]: #length of train data
         len(train_data)

Out[14]: 10000

In [15]: #length of test data
         len(test_data)

Out[15]: 1000
```

Q4. Print the first record in train data and print the first record as scentences with punctuation marks, separate the question and the answer

```
In [16]: # first entry in the train data
         train_data[0]

Out[16]: (['Mary',
           'moved',
           'to',
           'the',
           'bathroom',
           '.',
           'Sandra',
           'journeyed',
           'to',
           'the',
           'bedroom',
           '.'],
          ['Is', 'Sandra', 'in', 'the', 'hallway', '?'],
          'no')

In [17]: #Separate the question
         ' '.join(train_data[0][1])

Out[17]: 'Is Sandra in the hallway ?'

In [18]: #separate the Answer
         train_data[0][2]

Out[18]: 'no'
```

Setting up Vocabulary of All Words

Q5. fill the code where ever required

In [19]:

```
vocab = set()
```

In [20]:

```
#concatenate the train and test data as 'all_data'
all_data = test_data + train_data

for story, question , answer in all_data:
    vocab = vocab.union(set(story))
    vocab = vocab.union(set(question))
```

In [21]:

```
for story, question , answer in all_data:
    vocab = vocab.union(set(story))
    vocab = vocab.union(set(question))
```

In [22]:

```
vocab.add('no')
vocab.add('yes')
```

In [23]:

```
vocab
```

Out[23]:
```
{'.',
 '?',
 'Daniel',
 'Is',
 'John',
 'Mary',
 'Sandra',
 'apple',
 'back',
 'bathroom',
 'bedroom',
 'discarded',
 'down',
 'dropped',
 'football',
 'garden',
 'got',
 'grabbed',
 'hallway',
 'in',
 'journeyed',
 'kitchen',
 'left',
 'milk',
 'moved',
 'no',
 'office',
 'picked',
 'put',
 'the',
 'there',
 'to',
 'took',
 'travelled',
 'up',
 'went',
 'yes'}
```

In [24]:
```
vocab_len = len(vocab) + 1 #we add an extra space to hold a 0 for Keras's pa
```

In [25]:
```
vocab_len
```

Out[25]:
```
38
```

In [26]:
```
# find the maximum story length
all_story_lens = [len(data[0]) for data in all_data]
max_story_len = (max(all_story_lens))
max_story_len
```

Out[26]:
```
156
```

In [27]:
```
# find the maximum question length
max_question_len = max([len(data[1]) for data in all_data])
max_question_len
```

Out[27]:
```
6
```

Vectorizing the Data

Q6 Vectorize and tokenize the data

In [28]:
```
# Reserve 0 for pad_sequences
vocab_size = len(vocab) + 1
```

In [29]:
```
from keras_preprocessing.sequence import pad_sequences
from keras_preprocessing.text import Tokenizer
```

In [30]:
```
# integer encode sequences of words
tokenizer = Tokenizer(filters=[])
tokenizer.fit_on_texts(vocab)
```

In [31]:
```
word_index = tokenizer.word_index
word_index
```

Out[31]:

```
{'got': 1,
 'to': 2,
 'went': 3,
 'milk': 4,
 'journeyed': 5,
 'john': 6,
 'is': 7,
 'down': 8,
 'kitchen': 9,
 'garden': 10,
 'took': 11,
 'bathroom': 12,
 'back': 13,
 '?': 14,
 'sandra': 15,
 'dropped': 16,
 'office': 17,
 'football': 18,
 'travelled': 19,
 'no': 20,
 'there': 21,
 'yes': 22,
 'the': 23,
 'left': 24,
 'grabbed': 25,
 'discarded': 26,
 'mary': 27,
 'put': 28,
 'apple': 29,
 'up': 30,
 'daniel': 31,
 'moved': 32,
 '.': 33,
 'bedroom': 34,
 'in': 35,
 'picked': 36,
 'hallway': 37}
```

In [32]:

```python
train_story_text = []
train_question_text = []
train_answers = []

for story,question,answer in train_data:
    train_story_text.append(story)
    train_question_text.append(question)
```

In [33]:

```python
train_story_seq = tokenizer.texts_to_sequences(train_story_text)
```

In [34]:

```python
len(train_story_text)
```

Out[34]:

```
10000
```

In [35]:

```python
len(train_story_seq)
```

```
Out[35]: 10000
```

Functionalize Vectorization

```
In [36]: def vectorize_stories(data, word_index=tokenizer.word_index, max_story_len=m
             '''
             INPUT:

             data: consisting of Stories,Queries,and Answers
             word_index: word index dictionary from tokenizer
             max_story_len: the length of the longest story (used for pad_sequences f
             max_question_len: length of the longest question (used for pad_sequences


             OUTPUT:

             Vectorizes the stories,questions, and answers into padded sequences. We
             answer in the data. Then we convert the raw words to an word index value
             output list. Then once we have converted the words to numbers, we pad th

             Returns this in the form of a tuple (X,Xq,Y) (padded based on max length
             '''



             # X = STORIES
             X = []
             # Xq = QUERY/QUESTION
             Xq = []
             # Y = CORRECT ANSWER
             Y = []


             for story, query, answer in data:

                 # Grab the word index for every word in story
                 x = [word_index[word.lower()] for word in story]
                 # Grab the word index for every word in query
                 xq = [word_index[word.lower()] for word in query]

                 # Grab the Answers (either Yes/No so we don't need to use list compr
                 # Index 0 is reserved so we're going to use + 1
                 y = np.zeros(len(word_index) + 1)

                 # Now that y is all zeros and we know its just Yes/No , we can use n
                 #
                 y[word_index[answer]] = 1

                 # Append each set of story,query, and answer to their respective hol
                 X.append(x)
                 Xq.append(xq)
                 Y.append(y)

             # Finally, pad the sequences based on their max length so the RNN can be
```

```
    # RETURN TUPLE FOR UNPACKING
    return (pad_sequences(X, maxlen=max_story_len),pad_sequences(Xq, maxlen=
```

In [37]:
```
inputs_train, queries_train, answers_train = vectorize_stories(train_data)
```

In [38]:
```
inputs_test, queries_test, answers_test = vectorize_stories(test_data)
```

In [39]:
```
inputs_test
```

Out[39]:
```
array([[ 0,  0,  0, ..., 23, 34, 33],
       [ 0,  0,  0, ..., 23, 10, 33],
       [ 0,  0,  0, ..., 23, 10, 33],
       ...,
       [ 0,  0,  0, ..., 23, 29, 33],
       [ 0,  0,  0, ..., 23, 10, 33],
       [ 0,  0,  0, ..., 29, 21, 33]], dtype=int32)
```

In [40]:
```
queries_test
```

Out[40]:
```
array([[ 7,  6, 35, 23,  9, 14],
       [ 7,  6, 35, 23,  9, 14],
       [ 7,  6, 35, 23, 10, 14],
       ...,
       [ 7, 27, 35, 23, 34, 14],
       [ 7, 15, 35, 23, 10, 14],
       [ 7, 27, 35, 23, 10, 14]], dtype=int32)
```

In [41]:
```
answers_test
```

Out[41]:
```
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.]])
```

In [42]:
```
sum(answers_test)
```

Out[42]:
```
array([  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 503.,   0.,
       497.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.])
```

In [43]:
```
tokenizer.word_index['yes']
```

Out[43]:
```
22
```

In [44]:
```
tokenizer.word_index['no']
```

Out[44]:
```
20
```

Creating the Model

In [45]:

```
from keras.models import Sequential, Model
from keras.layers import Embedding
from keras.layers import Input, Activation, Dense, Permute, Dropout
from keras.layers import add, dot, concatenate
from keras.layers import LSTM
```

Placeholders for Inputs

Recall we technically have two inputs, stories and questions. So we need to use placeholders. `Input()` is used to instantiate a Keras tensor.

In [46]:

```
input_sequence = Input((max_story_len,))
question = Input((max_question_len,))
```

Building the Networks

To understand why we chose this setup, make sure to read the paper we are using:

- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, "End-To-End Memory Networks", http://arxiv.org/abs/1503.08895

Encoders

Q7 Create the layers as per the instructions given in the problem statement

Input Encoder m

In [47]:

```
#### Input gets embedded to a sequence of vectors
input_encoder_m = Sequential()
input_encoder_m.add(Embedding(input_dim=vocab_len,output_dim = 64)) #From pa
input_encoder_m.add(Dropout(0.3))
####This encoder will output:
#### (samples, story_maxlen, embedding_dim)
```

```
2023-03-21 00:36:31.730173: I tensorflow/core/platform/cpu_feature_guard.c
c:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

Input Encoder c

In [48]:

```
# embed the input into a sequence of vectors of size query_maxlen
# Follow the instructions in the problem statement
input_encoder_c = Sequential()
input_encoder_c.add(Embedding(input_dim=vocab_len,output_dim = max_question_
input_encoder_c.add(Dropout(0.3))
```

```
# output: (samples, story_maxlen, query_maxlen)
```

Question Encoder

In [49]:

```
# embed the question into a sequence of vectors
# Follow the instructions in the problem statement
# output: (samples, query_maxlen, embedding_dim)
question_encoder = Sequential()
question_encoder.add(Embedding(input_dim=vocab_len,output_dim = 64,input_ler
question_encoder.add(Dropout(0.3))
```

Encode the Sequences

In [50]:

```
# encode input sequence and questions (which are indices)
# to sequences of dense vectors
input_encoded_m = input_encoder_m(input_sequence)
input_encoded_c = input_encoder_c(input_sequence)
question_encoded = question_encoder(question)
```

Use dot product to compute the match between first input vector seq and the query

In [51]:

```
# shape: `(samples, story_maxlen, query_maxlen)`
match = dot([input_encoded_m, question_encoded], axes=(2, 2))
match = Activation('softmax')(match)
```

Add this match matrix with the second input vector sequence

In [52]:

```
# add the match matrix with the second input vector sequence
response = add([match, input_encoded_c])  # (samples, story_maxlen, query_ma
response = Permute((2, 1))(response)  # (samples, query_maxlen, story_maxler
```

Concatenate

In [53]:

```
# concatenate the match matrix with the question vector sequence
answer = concatenate([response, question_encoded])
```

In [54]:

```
answer
```

Out[54]:

```
<KerasTensor: shape=(None, 6, 220) dtype=float32 (created by layer 'concate
nate')>
```

In [55]:

```
# Reduce with RNN (LSTM)
answer = LSTM(32)(answer)  # (samples, 32)
```

In [56]:

```
# Regularization with Dropout
answer = Dropout(0.5)(answer)
answer = Dense(vocab_size)(answer)  # (samples, vocab_size)
```

In [57]:

```
# we output a probability distribution over the vocabulary
answer = Activation('softmax')(answer)

# build the final model
model = Model([input_sequence, question], answer)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [58]:

```
model.summary()
```

```
Model: "model"
___________________________________________________________________________
_______________________
 Layer (type)                   Output Shape         Param #     Connected
to
===========================================================================
=======================
 input_1 (InputLayer)           [(None, 156)]        0           []

 input_2 (InputLayer)           [(None, 6)]          0           []

 sequential (Sequential)        (None, None, 64)     2432        ['input_1
[0][0]']

 sequential_2 (Sequential)      (None, 6, 64)        2432        ['input_2
[0][0]']

 dot (Dot)                      (None, 156, 6)       0           ['sequenti
al[0][0]',
                                                                  'sequenti

al_2[0][0]']

 activation (Activation)        (None, 156, 6)       0           ['dot[0]
[0]']

 sequential_1 (Sequential)      (None, None, 6)      228         ['input_1
[0][0]']

 add (Add)                      (None, 156, 6)       0           ['activati
on[0][0]',
                                                                  'sequenti

al_1[0][0]']

 permute (Permute)              (None, 6, 156)       0           ['add[0]
[0]']

 concatenate (Concatenate)      (None, 6, 220)       0           ['permute
[0][0]',
                                                                  'sequenti

al_2[0][0]']

 lstm (LSTM)                    (None, 32)           32384       ['concaten
ate[0][0]']

 dropout_3 (Dropout)            (None, 32)           0           ['lstm[0]
[0]']

 dense (Dense)                  (None, 38)           1254        ['dropout_
3[0][0]']

 activation_1 (Activation)      (None, 38)           0           ['dense[0]
[0]']

===========================================================================
=======================
Total params: 38,730
```

```
Trainable params: 38,730
Non-trainable params: 0
___________________________________________________________________________
_______________________
```

In [59]:

```
# train
history = model.fit([inputs_train, queries_train], answers_train,batch_size=
Epoch 1/120
313/313 [==============================] - 5s 9ms/step - loss: 0.8962 - acc
uracy: 0.5016 - val_loss: 0.6951 - val_accuracy: 0.5030
Epoch 2/120
313/313 [==============================] - 3s 8ms/step - loss: 0.7068 - acc
uracy: 0.4928 - val_loss: 0.6951 - val_accuracy: 0.4970
Epoch 3/120
313/313 [==============================] - 3s 10ms/step - loss: 0.6980 - ac
curacy: 0.5099 - val_loss: 0.6932 - val_accuracy: 0.5030
Epoch 4/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6966 - acc
uracy: 0.4959 - val_loss: 0.6947 - val_accuracy: 0.5030
Epoch 5/120
313/313 [==============================] - 3s 10ms/step - loss: 0.6956 - ac
curacy: 0.5004 - val_loss: 0.6938 - val_accuracy: 0.4970
Epoch 6/120
313/313 [==============================] - 3s 10ms/step - loss: 0.6959 - ac
curacy: 0.4941 - val_loss: 0.6950 - val_accuracy: 0.4970
Epoch 7/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6954 - acc
uracy: 0.4970 - val_loss: 0.6933 - val_accuracy: 0.5030
Epoch 8/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6955 - acc
uracy: 0.4983 - val_loss: 0.6932 - val_accuracy: 0.4970
Epoch 9/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6956 - acc
uracy: 0.4881 - val_loss: 0.6932 - val_accuracy: 0.5030
Epoch 10/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6957 - acc
uracy: 0.4890 - val_loss: 0.6944 - val_accuracy: 0.4970
Epoch 11/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6954 - acc
uracy: 0.4934 - val_loss: 0.6931 - val_accuracy: 0.5030
Epoch 12/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6942 - acc
uracy: 0.5066 - val_loss: 0.6953 - val_accuracy: 0.5030
Epoch 13/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.4982 - val_loss: 0.6957 - val_accuracy: 0.4970
Epoch 14/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6957 - acc
uracy: 0.4974 - val_loss: 0.6932 - val_accuracy: 0.5030
Epoch 15/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6946 - acc
uracy: 0.5037 - val_loss: 0.6933 - val_accuracy: 0.5030
Epoch 16/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6953 - acc
uracy: 0.5001 - val_loss: 0.6938 - val_accuracy: 0.5030
Epoch 17/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6954 - acc
uracy: 0.4908 - val_loss: 0.6933 - val_accuracy: 0.4970
Epoch 18/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6949 - acc
uracy: 0.5037 - val_loss: 0.6933 - val_accuracy: 0.5030
Epoch 19/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6953 - acc
```

```
uracy: 0.5004 - val_loss: 0.6941 - val_accuracy: 0.4970
Epoch 20/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6944 - acc
uracy: 0.5050 - val_loss: 0.6933 - val_accuracy: 0.4970
Epoch 21/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6945 - acc
uracy: 0.5082 - val_loss: 0.6934 - val_accuracy: 0.4970
Epoch 22/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6954 - acc
uracy: 0.4927 - val_loss: 0.6948 - val_accuracy: 0.4970
Epoch 23/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6953 - acc
uracy: 0.4983 - val_loss: 0.6935 - val_accuracy: 0.4970
Epoch 24/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6949 - acc
uracy: 0.4991 - val_loss: 0.6932 - val_accuracy: 0.5030
Epoch 25/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6949 - acc
uracy: 0.4928 - val_loss: 0.6955 - val_accuracy: 0.4970
Epoch 26/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.5032 - val_loss: 0.6972 - val_accuracy: 0.4970
Epoch 27/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6947 - acc
uracy: 0.5012 - val_loss: 0.6935 - val_accuracy: 0.4970
Epoch 28/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6945 - acc
uracy: 0.5021 - val_loss: 0.6935 - val_accuracy: 0.5030
Epoch 29/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.4958 - val_loss: 0.6931 - val_accuracy: 0.5030
Epoch 30/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.5013 - val_loss: 0.6933 - val_accuracy: 0.4970
Epoch 31/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6941 - acc
uracy: 0.5075 - val_loss: 0.6936 - val_accuracy: 0.5030
Epoch 32/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6950 - acc
uracy: 0.4947 - val_loss: 0.6931 - val_accuracy: 0.5030
Epoch 33/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6950 - acc
uracy: 0.4967 - val_loss: 0.6942 - val_accuracy: 0.5030
Epoch 34/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6953 - acc
uracy: 0.4952 - val_loss: 0.6933 - val_accuracy: 0.4970
Epoch 35/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6946 - acc
uracy: 0.4964 - val_loss: 0.6931 - val_accuracy: 0.5030
Epoch 36/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6948 - acc
uracy: 0.5021 - val_loss: 0.6953 - val_accuracy: 0.4970
Epoch 37/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.4971 - val_loss: 0.6933 - val_accuracy: 0.5030
Epoch 38/120
```

```
313/313 [==============================] - 3s 8ms/step - loss: 0.6952 - acc
uracy: 0.5030 - val_loss: 0.6934 - val_accuracy: 0.4970
Epoch 39/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6947 - acc
uracy: 0.4981 - val_loss: 0.6932 - val_accuracy: 0.5030
Epoch 40/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6948 - acc
uracy: 0.4994 - val_loss: 0.6942 - val_accuracy: 0.5030
Epoch 41/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6942 - acc
uracy: 0.5071 - val_loss: 0.6954 - val_accuracy: 0.5030
Epoch 42/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6956 - acc
uracy: 0.4952 - val_loss: 0.6932 - val_accuracy: 0.5030
Epoch 43/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.4929 - val_loss: 0.6937 - val_accuracy: 0.4970
Epoch 44/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6950 - acc
uracy: 0.4961 - val_loss: 0.6937 - val_accuracy: 0.4970
Epoch 45/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6940 - acc
uracy: 0.5056 - val_loss: 0.6932 - val_accuracy: 0.4910
Epoch 46/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6951 - acc
uracy: 0.5010 - val_loss: 0.6932 - val_accuracy: 0.4830
Epoch 47/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6949 - acc
uracy: 0.4985 - val_loss: 0.6933 - val_accuracy: 0.4810
Epoch 48/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6948 - acc
uracy: 0.4995 - val_loss: 0.6937 - val_accuracy: 0.4970
Epoch 49/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6946 - acc
uracy: 0.5034 - val_loss: 0.6935 - val_accuracy: 0.5020
Epoch 50/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6945 - acc
uracy: 0.5044 - val_loss: 0.6947 - val_accuracy: 0.4970
Epoch 51/120
313/313 [==============================] - 2s 8ms/step - loss: 0.6950 - acc
uracy: 0.4944 - val_loss: 0.6945 - val_accuracy: 0.5010
Epoch 52/120
313/313 [==============================] - 2s 7ms/step - loss: 0.6938 - acc
uracy: 0.5156 - val_loss: 0.6954 - val_accuracy: 0.4790
Epoch 53/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6936 - acc
uracy: 0.5146 - val_loss: 0.6952 - val_accuracy: 0.5020
Epoch 54/120
313/313 [==============================] - 3s 11ms/step - loss: 0.6941 - ac
curacy: 0.5118 - val_loss: 0.6961 - val_accuracy: 0.4840
Epoch 55/120
313/313 [==============================] - 3s 10ms/step - loss: 0.6939 - ac
curacy: 0.5086 - val_loss: 0.6954 - val_accuracy: 0.4780
Epoch 56/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6924 - acc
uracy: 0.5236 - val_loss: 0.6977 - val_accuracy: 0.4840
```

```
Epoch 57/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6937 - acc
uracy: 0.5102 - val_loss: 0.6965 - val_accuracy: 0.4750
Epoch 58/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6934 - acc
uracy: 0.5205 - val_loss: 0.6965 - val_accuracy: 0.4770
Epoch 59/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6922 - acc
uracy: 0.5211 - val_loss: 0.6955 - val_accuracy: 0.4900
Epoch 60/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6899 - acc
uracy: 0.5389 - val_loss: 0.6968 - val_accuracy: 0.4920
Epoch 61/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6859 - acc
uracy: 0.5413 - val_loss: 0.6950 - val_accuracy: 0.5360
Epoch 62/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6799 - acc
uracy: 0.5610 - val_loss: 0.6812 - val_accuracy: 0.5580
Epoch 63/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6692 - acc
uracy: 0.5781 - val_loss: 0.6665 - val_accuracy: 0.5690
Epoch 64/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6597 - acc
uracy: 0.5994 - val_loss: 0.6549 - val_accuracy: 0.6050
Epoch 65/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6519 - acc
uracy: 0.6160 - val_loss: 0.6432 - val_accuracy: 0.6490
Epoch 66/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6411 - acc
uracy: 0.6367 - val_loss: 0.6386 - val_accuracy: 0.6290
Epoch 67/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6337 - acc
uracy: 0.6469 - val_loss: 0.6242 - val_accuracy: 0.6600
Epoch 68/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6289 - acc
uracy: 0.6514 - val_loss: 0.6275 - val_accuracy: 0.6550
Epoch 69/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6253 - acc
uracy: 0.6595 - val_loss: 0.6248 - val_accuracy: 0.6600
Epoch 70/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6220 - acc
uracy: 0.6554 - val_loss: 0.6173 - val_accuracy: 0.6650
Epoch 71/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6178 - acc
uracy: 0.6649 - val_loss: 0.6190 - val_accuracy: 0.6630
Epoch 72/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6136 - acc
uracy: 0.6597 - val_loss: 0.6110 - val_accuracy: 0.6680
Epoch 73/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6137 - acc
uracy: 0.6721 - val_loss: 0.6231 - val_accuracy: 0.6440
Epoch 74/120
313/313 [==============================] - 3s 9ms/step - loss: 0.6075 - acc
uracy: 0.6716 - val_loss: 0.6130 - val_accuracy: 0.6700
Epoch 75/120
313/313 [==============================] - 3s 8ms/step - loss: 0.6010 - acc
```

```
uracy: 0.6759 - val_loss: 0.5954 - val_accuracy: 0.6830
Epoch 76/120
313/313 [==============================] - 3s 9ms/step - loss: 0.5894 - acc
uracy: 0.6878 - val_loss: 0.5964 - val_accuracy: 0.6750
Epoch 77/120
313/313 [==============================] - 3s 9ms/step - loss: 0.5785 - acc
uracy: 0.6990 - val_loss: 0.5734 - val_accuracy: 0.7030
Epoch 78/120
313/313 [==============================] - 3s 9ms/step - loss: 0.5642 - acc
uracy: 0.7061 - val_loss: 0.5587 - val_accuracy: 0.7110
Epoch 79/120
313/313 [==============================] - 3s 9ms/step - loss: 0.5536 - acc
uracy: 0.7168 - val_loss: 0.5478 - val_accuracy: 0.7190
Epoch 80/120
313/313 [==============================] - 3s 9ms/step - loss: 0.5399 - acc
uracy: 0.7283 - val_loss: 0.5434 - val_accuracy: 0.7180
Epoch 81/120
313/313 [==============================] - 3s 8ms/step - loss: 0.5274 - acc
uracy: 0.7334 - val_loss: 0.5333 - val_accuracy: 0.7290
Epoch 82/120
313/313 [==============================] - 3s 8ms/step - loss: 0.5183 - acc
uracy: 0.7433 - val_loss: 0.5083 - val_accuracy: 0.7440
Epoch 83/120
313/313 [==============================] - 3s 9ms/step - loss: 0.5051 - acc
uracy: 0.7508 - val_loss: 0.4962 - val_accuracy: 0.7670
Epoch 84/120
313/313 [==============================] - 3s 10ms/step - loss: 0.4842 - ac
curacy: 0.7724 - val_loss: 0.4917 - val_accuracy: 0.7700
Epoch 85/120
313/313 [==============================] - 3s 10ms/step - loss: 0.4665 - ac
curacy: 0.7873 - val_loss: 0.4753 - val_accuracy: 0.7820
Epoch 86/120
313/313 [==============================] - 3s 9ms/step - loss: 0.4517 - acc
uracy: 0.7942 - val_loss: 0.4400 - val_accuracy: 0.7970
Epoch 87/120
313/313 [==============================] - 3s 9ms/step - loss: 0.4431 - acc
uracy: 0.8022 - val_loss: 0.4392 - val_accuracy: 0.7990
Epoch 88/120
313/313 [==============================] - 3s 8ms/step - loss: 0.4201 - acc
uracy: 0.8159 - val_loss: 0.4298 - val_accuracy: 0.7950
Epoch 89/120
313/313 [==============================] - 3s 8ms/step - loss: 0.4072 - acc
uracy: 0.8233 - val_loss: 0.4067 - val_accuracy: 0.8180
Epoch 90/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3993 - acc
uracy: 0.8296 - val_loss: 0.4151 - val_accuracy: 0.8170
Epoch 91/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3921 - acc
uracy: 0.8329 - val_loss: 0.4002 - val_accuracy: 0.8200
Epoch 92/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3861 - acc
uracy: 0.8354 - val_loss: 0.4261 - val_accuracy: 0.8080
Epoch 93/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3792 - acc
uracy: 0.8376 - val_loss: 0.3940 - val_accuracy: 0.8310
Epoch 94/120
```

```
313/313 [==============================] - 3s 8ms/step - loss: 0.3706 - acc
uracy: 0.8443 - val_loss: 0.3966 - val_accuracy: 0.8270
Epoch 95/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3653 - acc
uracy: 0.8415 - val_loss: 0.4124 - val_accuracy: 0.8250
Epoch 96/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3601 - acc
uracy: 0.8473 - val_loss: 0.3958 - val_accuracy: 0.8220
Epoch 97/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3615 - acc
uracy: 0.8461 - val_loss: 0.4036 - val_accuracy: 0.8100
Epoch 98/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3572 - acc
uracy: 0.8506 - val_loss: 0.3836 - val_accuracy: 0.8310
Epoch 99/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3510 - acc
uracy: 0.8521 - val_loss: 0.4056 - val_accuracy: 0.7920
Epoch 100/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3468 - acc
uracy: 0.8526 - val_loss: 0.3846 - val_accuracy: 0.8300
Epoch 101/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3429 - acc
uracy: 0.8582 - val_loss: 0.3934 - val_accuracy: 0.8430
Epoch 102/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3439 - acc
uracy: 0.8555 - val_loss: 0.3806 - val_accuracy: 0.8440
Epoch 103/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3401 - acc
uracy: 0.8548 - val_loss: 0.3744 - val_accuracy: 0.8310
Epoch 104/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3323 - acc
uracy: 0.8604 - val_loss: 0.3990 - val_accuracy: 0.8260
Epoch 105/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3354 - acc
uracy: 0.8588 - val_loss: 0.3876 - val_accuracy: 0.8370
Epoch 106/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3367 - acc
uracy: 0.8588 - val_loss: 0.3850 - val_accuracy: 0.8210
Epoch 107/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3300 - acc
uracy: 0.8622 - val_loss: 0.4203 - val_accuracy: 0.8120
Epoch 108/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3282 - acc
uracy: 0.8660 - val_loss: 0.3959 - val_accuracy: 0.8340
Epoch 109/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3236 - acc
uracy: 0.8647 - val_loss: 0.3903 - val_accuracy: 0.8330
Epoch 110/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3267 - acc
uracy: 0.8617 - val_loss: 0.3936 - val_accuracy: 0.8290
Epoch 111/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3184 - acc
uracy: 0.8666 - val_loss: 0.3948 - val_accuracy: 0.8330
Epoch 112/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3195 - acc
uracy: 0.8635 - val_loss: 0.3897 - val_accuracy: 0.8370
```

```
Epoch 113/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3232 - acc
uracy: 0.8670 - val_loss: 0.3828 - val_accuracy: 0.8380
Epoch 114/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3108 - acc
uracy: 0.8681 - val_loss: 0.3769 - val_accuracy: 0.8310
Epoch 115/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3173 - acc
uracy: 0.8677 - val_loss: 0.4001 - val_accuracy: 0.8180
Epoch 116/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3101 - acc
uracy: 0.8663 - val_loss: 0.4072 - val_accuracy: 0.8360
Epoch 117/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3101 - acc
uracy: 0.8680 - val_loss: 0.4002 - val_accuracy: 0.8340
Epoch 118/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3032 - acc
uracy: 0.8730 - val_loss: 0.3949 - val_accuracy: 0.8320
Epoch 119/120
313/313 [==============================] - 3s 8ms/step - loss: 0.3037 - acc
uracy: 0.8730 - val_loss: 0.3884 - val_accuracy: 0.8350
Epoch 120/120
313/313 [==============================] - 3s 9ms/step - loss: 0.3022 - acc
uracy: 0.8707 - val_loss: 0.3899 - val_accuracy: 0.8340
```

Saving the Model

In [60]:

```
filename = 'chatbot_120_epochs.h5'
model.save(filename)
```

Evaluating the Model

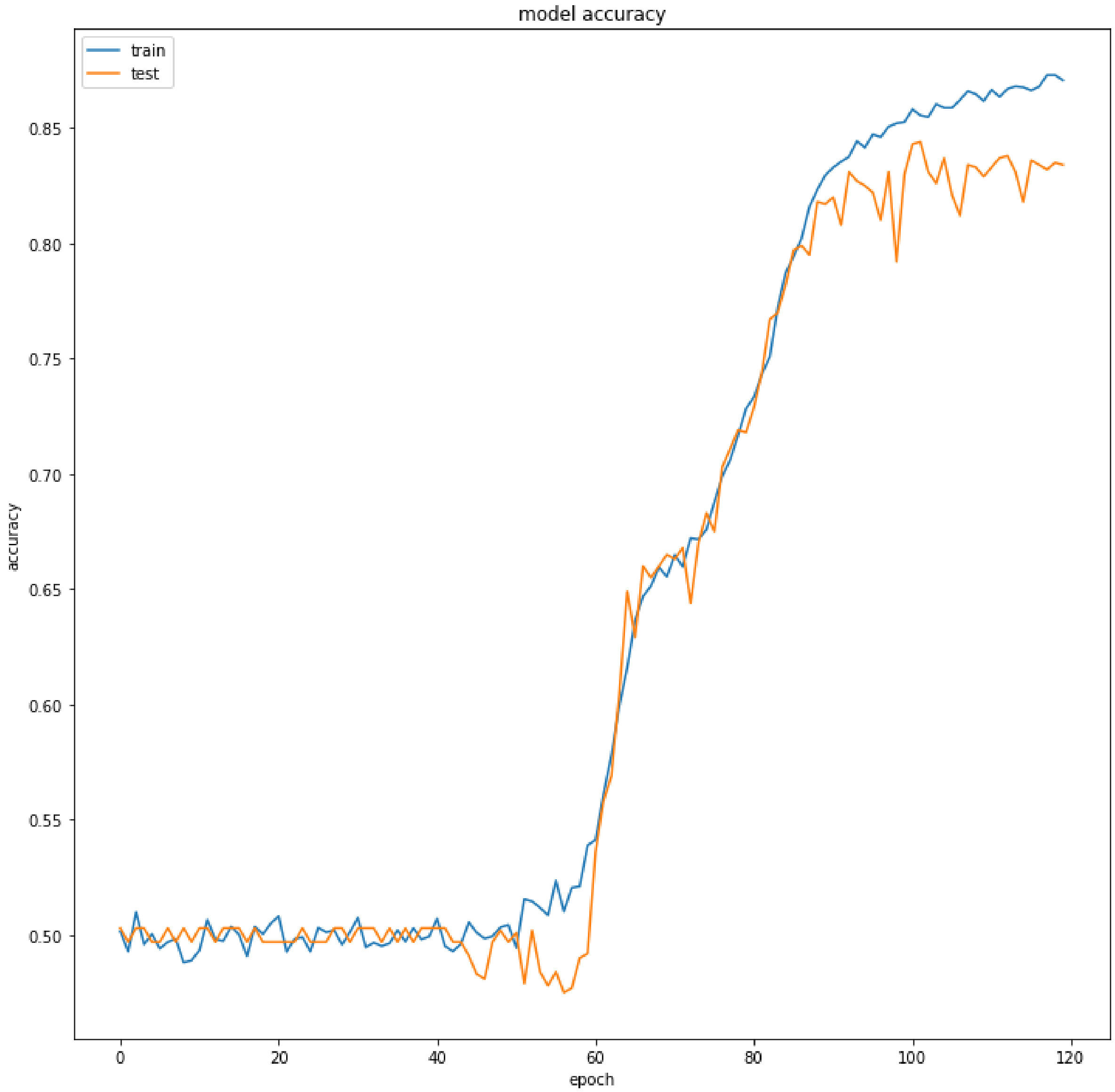
Plotting Out Training History

Q8 Write your code to plot the training history

In [61]:

```
import matplotlib.pyplot as plt
%matplotlib inline
print(history.history.keys())
plt.figure(figsize=(12,12))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Evaluating on Given Test Set

In [62]:
```
model.load_weights(filename)
pred_results = model.predict(([inputs_test, queries_test]))
```

```
32/32 [==============================] - 0s 3ms/step
```

In [63]:
```
test_data[0][0]
```

Out[63]:

```
['Mary',
 'got',
 'the',
 'milk',
 'there',
 '.',
 'John',
 'moved',
 'to',
 'the',
 'bedroom',
 '.']
```

In [64]:

```
story =' '.join(word for word in test_data[0][0])
print(story)
```

```
Mary got the milk there . John moved to the bedroom .
```

In [65]:

```
query = ' '.join(word for word in test_data[0][1])
print(query)
```

```
Is John in the kitchen ?
```

In [66]:

```
print("True Test Answer from Data is:",test_data[0][2])
```

```
True Test Answer from Data is: no
```

In [67]:

```
#Generate prediction from model
val_max = np.argmax(pred_results[0])

for key, val in tokenizer.word_index.items():
    if val == val_max:
        k = key

print("Predicted answer is: ", k)
print("Probability of certainty was: ", pred_results[0][val_max])
```

```
Predicted answer is:  no
Probability of certainty was:  0.9978237
```

Writing Your Own Stories and Questions

Remember you can only use words from the existing vocab

Q8 use the model for predicting the given strory and question given in the problem statement

In [68]:

```
q8_story="Daniel went up to bedroom . John dropped the football in the Kitch

q8_ask="Is the football in the Kitchen ?"

ask_data = [(q8_story.split(), q8_ask.split(),'no')]

ask_story, ask_ques, ask_ans = vectorize_stories(ask_data)
```

```
pred_results = model.predict(([ask_story,ask_ques]))
val_max = np.argmax(pred_results[0])
```

```
1/1 [==============================] - 0s 20ms/step
```

In [69]:

```
#Correct prediction!
for key,val in tokenizer.word_index.items():
    if val == val_max:
        k = key
print(k)
```

```
yes
```

In [70]:

```
pred_results[0][val_max]
```

Out[70]:

```
0.95275676
```

Great Job!

In [ ]:

```

```