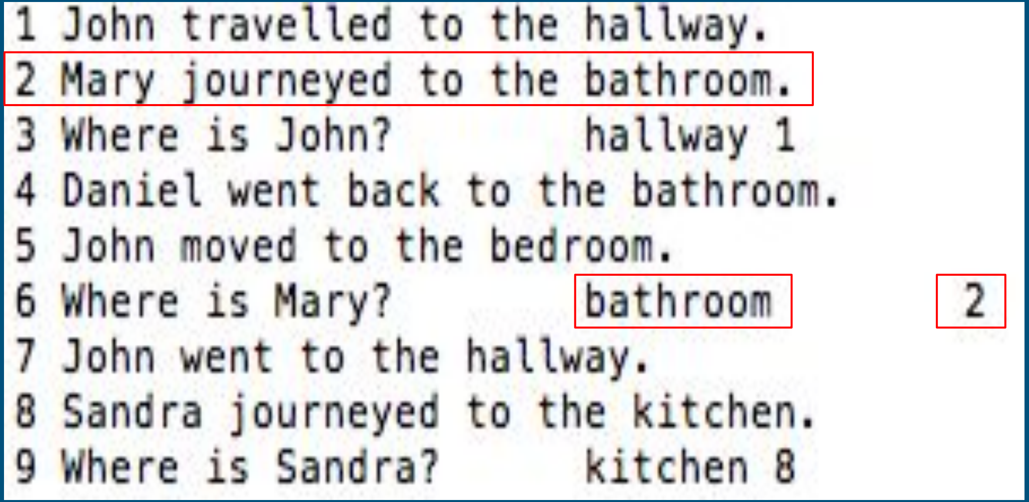
Question Answering on Reading Comprehension

Srinivasan Vasudevan,
Greg Murray, Kyle Riener,
Sofia Kenny, Elva Shen

Data

- ❖ Trained a network on the bAbi data set for reading comprehension.
- ❖ Contains 9000 questions for training and 1000 questions for testing
- ❖ Contains a set of contexts with multiple question answer pairs



```
1 John travelled to the hallway.
2 Mary journeyed to the bathroom.
3 Where is John?        hallway 1
4 Daniel went back to the bathroom.
5 John moved to the bedroom.
6 Where is Mary?        bathroom      2
7 John went to the hallway.
8 Sandra journeyed to the kitchen.
9 Where is Sandra?      kitchen 8
```

Source: https://research.fb.com/downloads/babi/

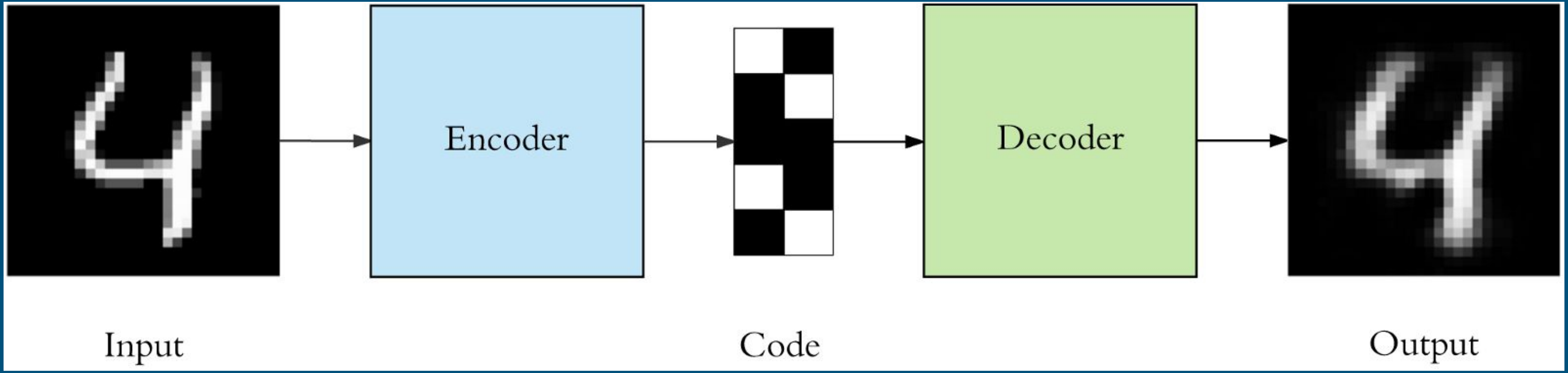
Data Preprocessing

- ❖ Retrieve the stories
- ❖ Parse stories
- ❖ Tokenize the sentences
- ❖ Vectorize stories

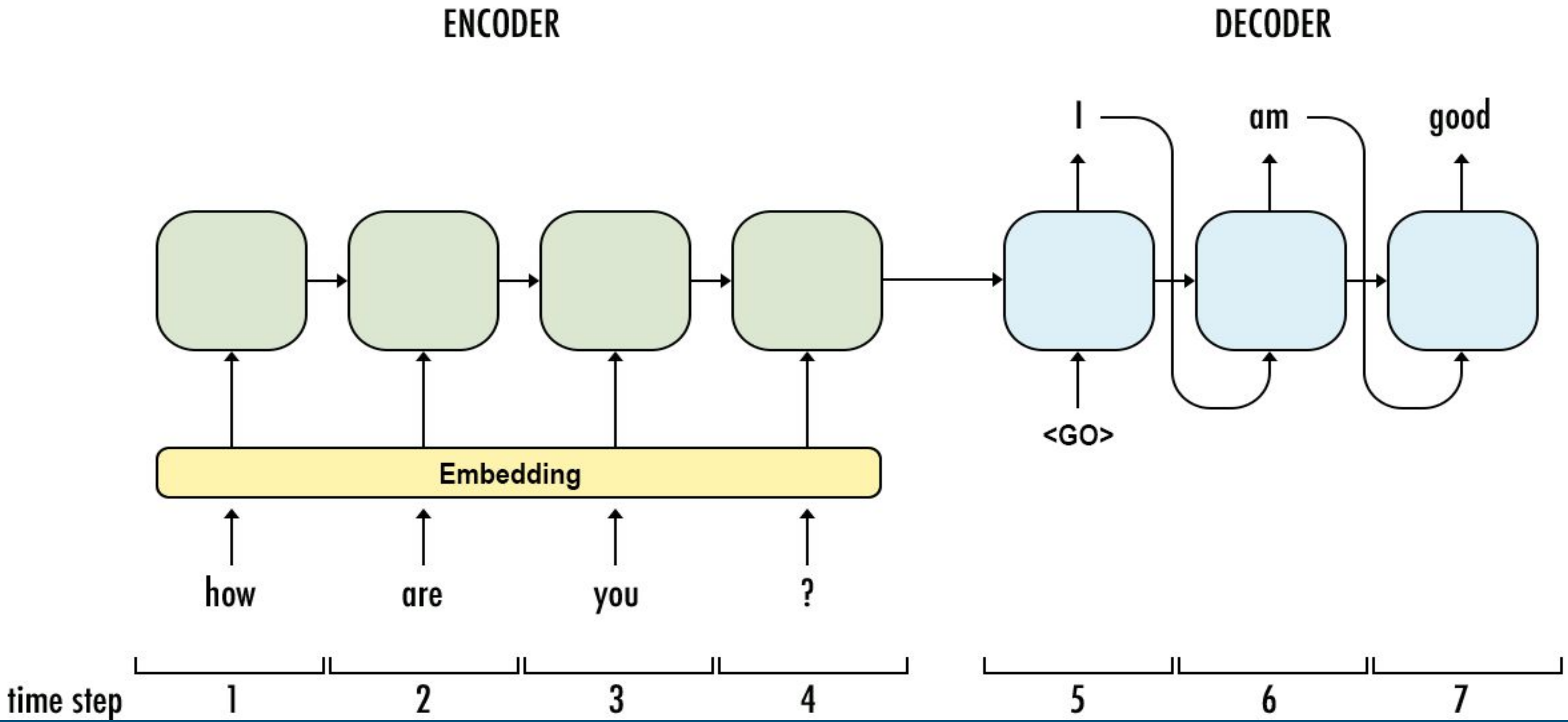
Auto-Encoder

- ❖ Consist of 3 components: encoder, code, decoder.
- ❖ Need 3 things: an encoding method, decoding method, and a loss function to compare the output with the target.

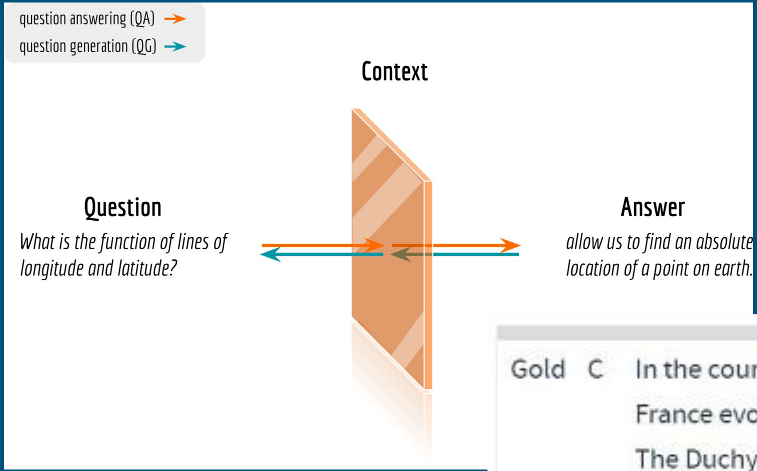
- ❖ For Dimensionality Reduction (Compression).

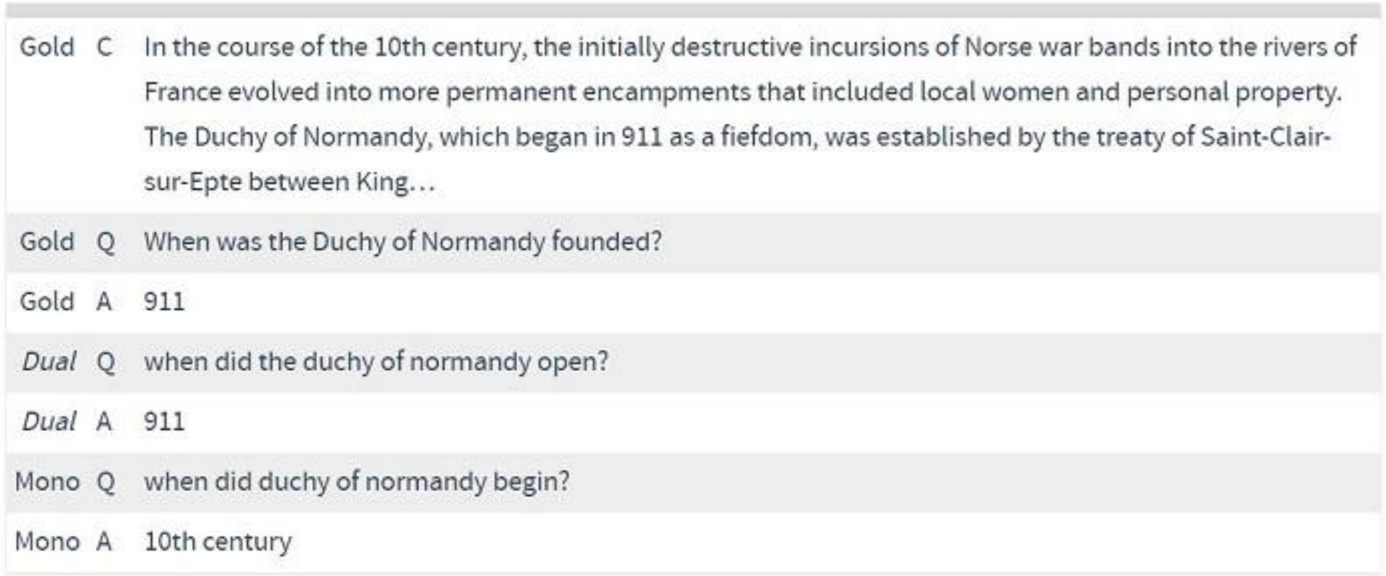


Sequential Encoding

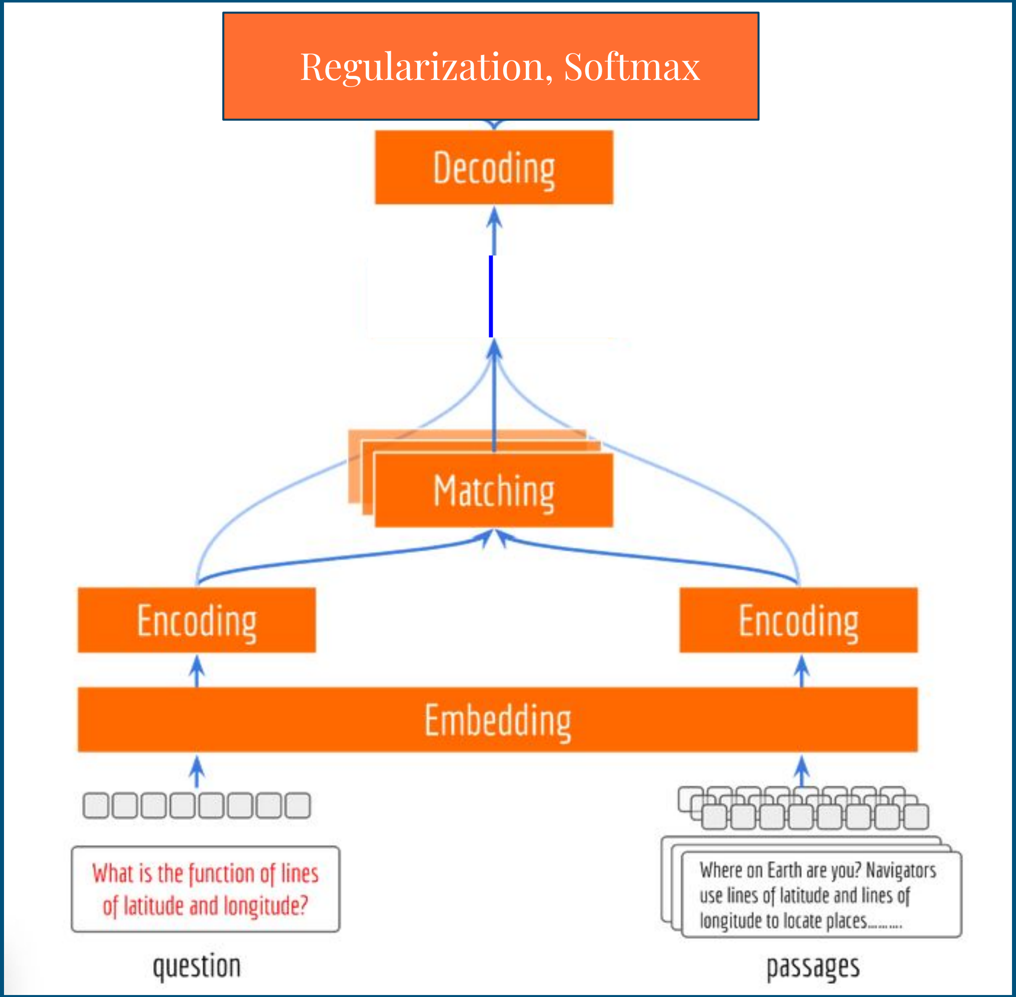


QA (Question Answering) Paradigm


question answering (QA)
question generation (QG)
Context
Question
What is the function of lines of longitude and latitude?
Answer
allow us to find an absolute location of a point on earth.



Gold	C	In the course of the 10th century, the initially destructive incursions of Norse war bands into the rivers of France evolved into more permanent encampments that included local women and personal property. The Duchy of Normandy, which began in 911 as a fiefdom, was established by the treaty of Saint-Clair-sur-Epte between King…
Gold	Q	When was the Duchy of Normandy founded?
Gold	A	911
Dual	Q	when did the duchy of normandy open?
Dual	A	911
Mono	Q	when did duchy of normandy begin?
Mono	A	10th century


Regularization, Softmax
Decoding
Matching
Encoding
Encoding
Embedding
What is the function of lines of latitude and longitude?
Where on Earth are you? Navigators use lines of latitude and lines of longitude to locate places.........
question
passages

- ❖ Input: Sequence of question and corresponding passage words as numeric vectors
- ❖ Embedding: convert sequence of token to tensors of fixed dimension (2D)
- ❖ Encoding: represent each word as numeric vector
- ❖ Decoding: convert vector back to word
- ❖ Matching: passage vectors are weighted according to their relations to the question

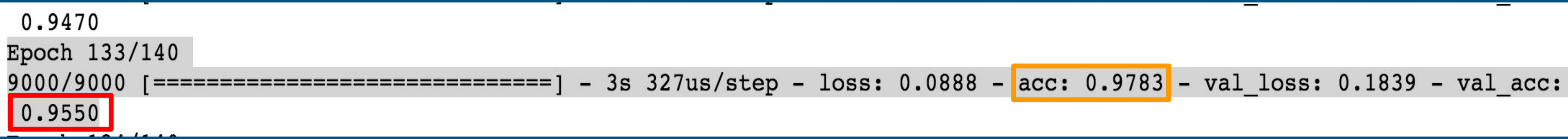
We present below two models: A simple model and a complex model.

Model Details (Simple Model)

- Used Keras Sequential Model.
- Encode question and context sequence.
- Embed question and context to 2D tensors.
- Create Match between input context and question
  - Perform dot product between encoded input and question
  - Obtain Matched output by performing softmax on the dot product result
  - Add Matched output (after softmax) and the encoded input context to amplify the matched output

Model Details - Simple Model (matching answer to question)

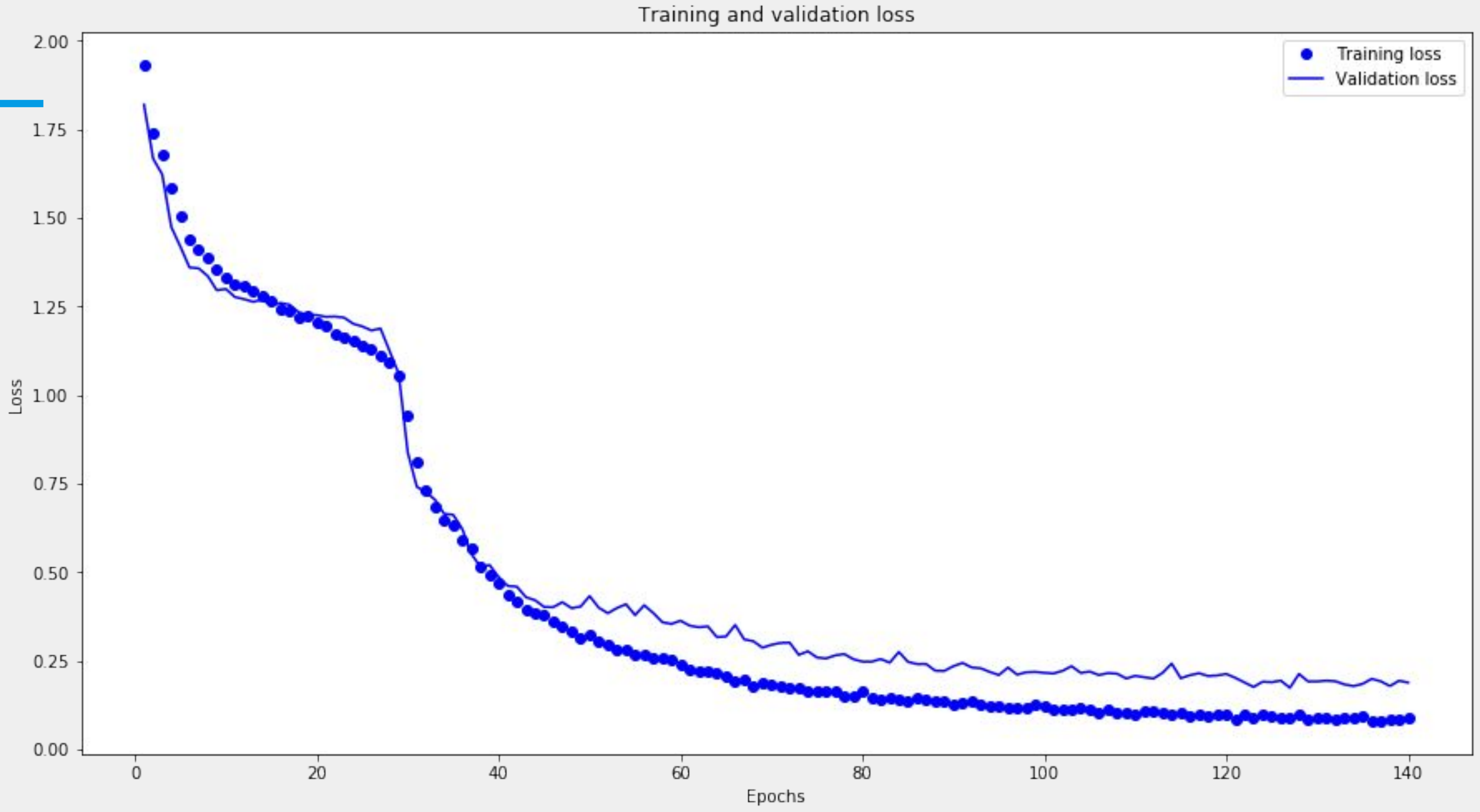
- Bidirectional LSTM (Decoder) with 64 units, dropout = 0.2
- Dense layer with L1 regularization, ReLU activation
- Dense layer with L2 regularization, softmax activation
- 140 epochs, Adam optimizer, categorical cross-entropy loss



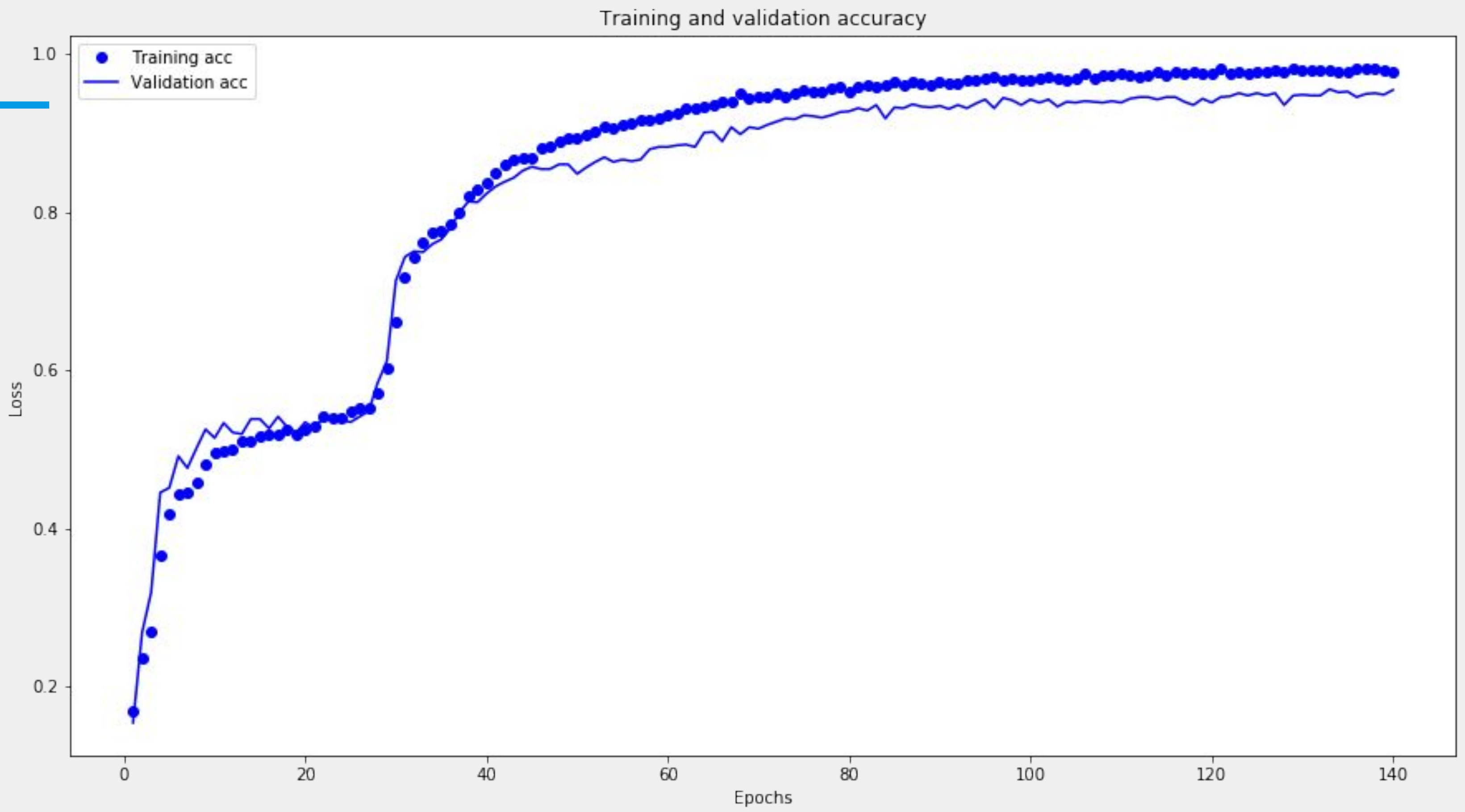
```
 0.9470
Epoch 133/140
9000/9000 [==============================] - 3s 327us/step - loss: 0.0888 - acc: 0.9783 - val_loss: 0.1839 - val_acc:
 0.9550
```

Code : Reading Comprehension Final Model.ipynb

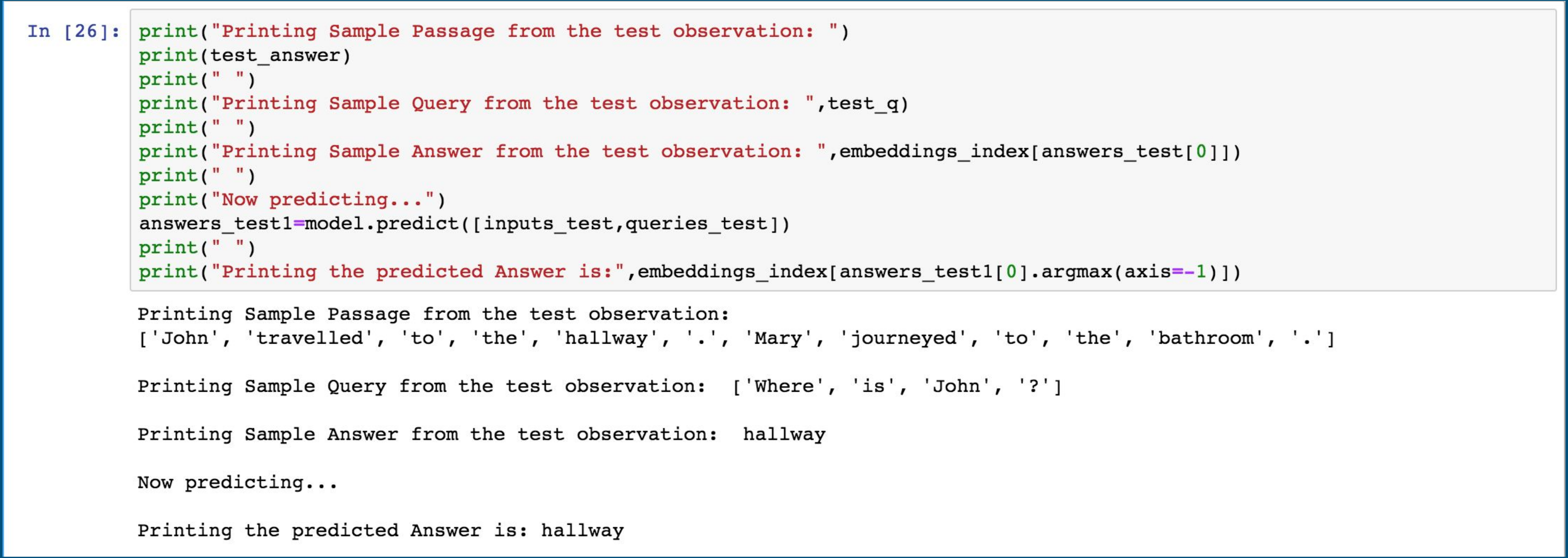
Epochs vs Training/Validation Losses



Epochs vs Training/Validation Accuracy



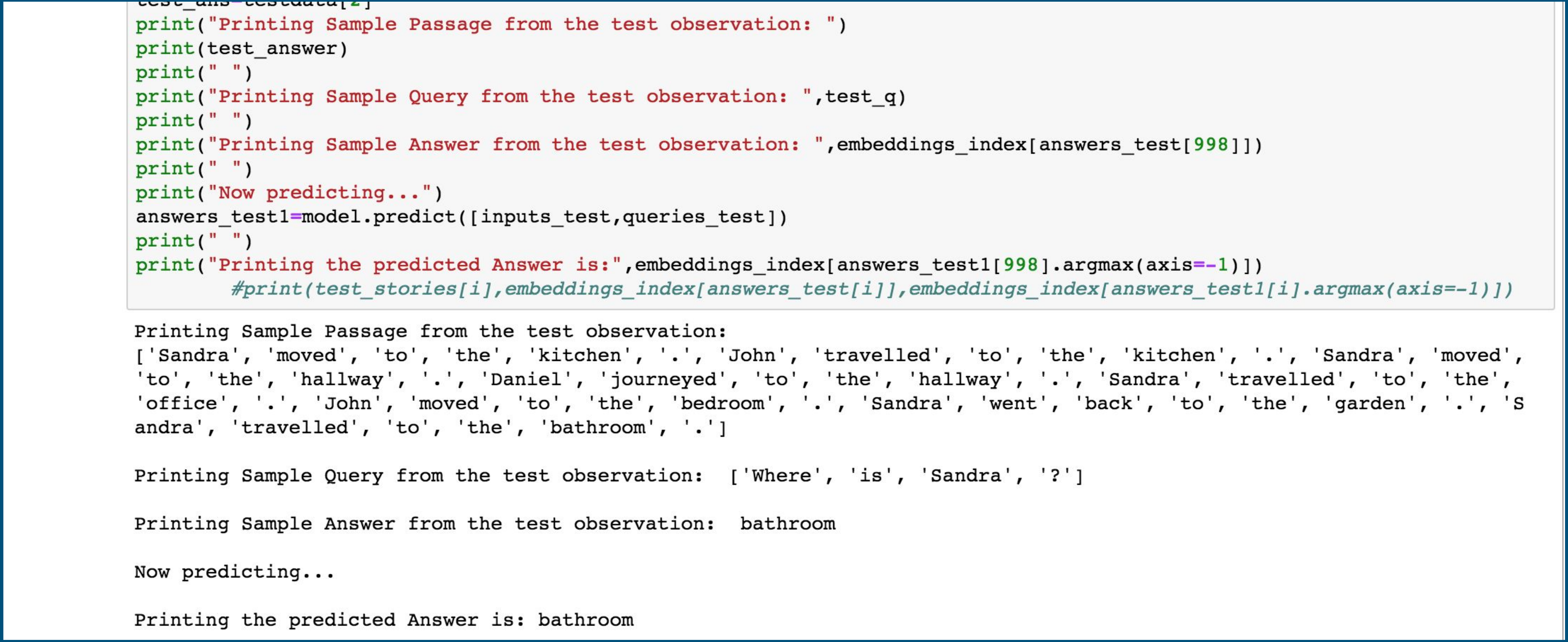
Test Sample Prediction (correct - Simple Case)



```
In [26]: print("Printing Sample Passage from the test observation: ")
         print(test_answer)
         print(" ")
         print("Printing Sample Query from the test observation: ",test_q)
         print(" ")
         print("Printing Sample Answer from the test observation: ",embeddings_index[answers_test[0]])
         print(" ")
         print("Now predicting...")
         answers_test1=model.predict([inputs_test,queries_test])
         print(" ")
         print("Printing the predicted Answer is:",embeddings_index[answers_test1[0].argmax(axis=-1)])
```

```
Printing Sample Passage from the test observation:
['John', 'travelled', 'to', 'the', 'hallway', '.', 'Mary', 'journeyed', 'to', 'the', 'bathroom', '.']

Printing Sample Query from the test observation:  ['Where', 'is', 'John', '?']

Printing Sample Answer from the test observation:  hallway

Now predicting...

Printing the predicted Answer is: hallway
```

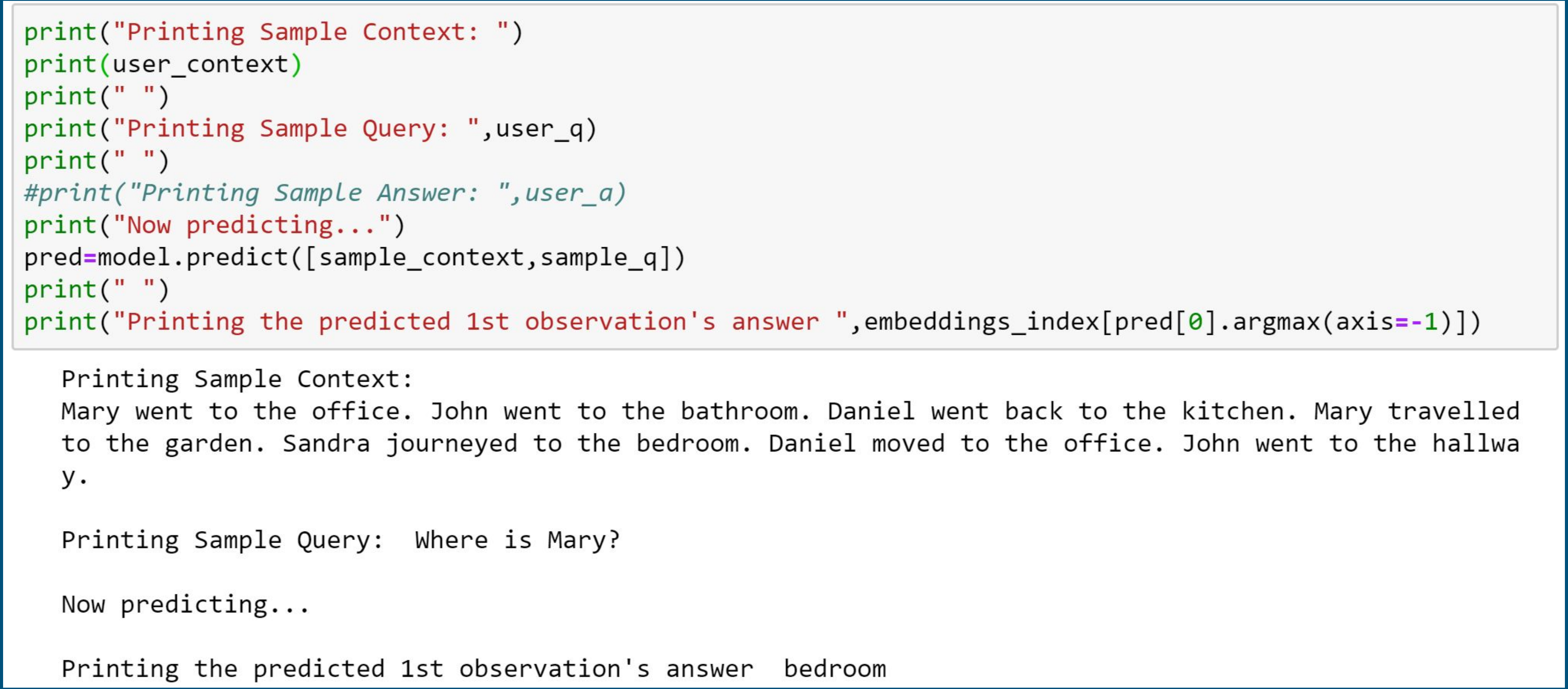
Test Sample Prediction (correct - Complex Case)



```
print("Printing Sample Passage from the test observation: ")
print(test_answer)
print(" ")
print("Printing Sample Query from the test observation: ",test_q)
print(" ")
print("Printing Sample Answer from the test observation: ",embeddings_index[answers_test[998]])
print(" ")
print("Now predicting...")
answers_test1=model.predict([inputs_test,queries_test])
print(" ")
print("Printing the predicted Answer is:",embeddings_index[answers_test1[998].argmax(axis=-1)])
        #print(test_stories[i],embeddings_index[answers_test[i]],embeddings_index[answers_test1[i].argmax(axis=-1)])
```

```
Printing Sample Passage from the test observation:
['Sandra', 'moved', 'to', 'the', 'kitchen', '.', 'John', 'travelled', 'to', 'the', 'kitchen', '.', 'Sandra', 'moved',
'to', 'the', 'hallway', '.', 'Daniel', 'journeyed', 'to', 'the', 'hallway', '.', 'Sandra', 'travelled', 'to', 'the',
'office', '.', 'John', 'moved', 'to', 'the', 'bedroom', '.', 'Sandra', 'went', 'back', 'to', 'the', 'garden', '.', 'S
andra', 'travelled', 'to', 'the', 'bathroom', '.']

Printing Sample Query from the test observation:  ['Where', 'is', 'Sandra', '?']

Printing Sample Answer from the test observation:  bathroom

Now predicting...

Printing the predicted Answer is: bathroom
```

Test Sample Prediction (wrong - Complex Case)



```
print("Printing Sample Context: ")
print(user_context)
print(" ")
print("Printing Sample Query: ",user_q)
print(" ")
#print("Printing Sample Answer: ",user_a)
print("Now predicting...")
pred=model.predict([sample_context,sample_q])
print(" ")
print("Printing the predicted 1st observation's answer ",embeddings_index[pred[0].argmax(axis=-1)])
```

```
Printing Sample Context:
Mary went to the office. John went to the bathroom. Daniel went back to the kitchen. Mary travelled
to the garden. Sandra journeyed to the bedroom. Daniel moved to the office. John went to the hallwa
y.

Printing Sample Query:  Where is Mary?

Now predicting...

Printing the predicted 1st observation's answer  bedroom
```

Limitations

- ❖ Simple, basic dataset
  - ➢ Examples are not very interesting or useful
- ❖ All words must be in the vocabulary
  - ➢ Struggles with names it has not seen before
  - ➢ Cannot identify simple synonyms, e.g. “strolled” instead of “walked”

Next Steps

- Experiment with other datasets
  - Examples: SQuAD, CNN/Daily Mail
- Experiment with network layout
  - Add layers after encoding
- Compare with pretrained models
  - Google’s BERT

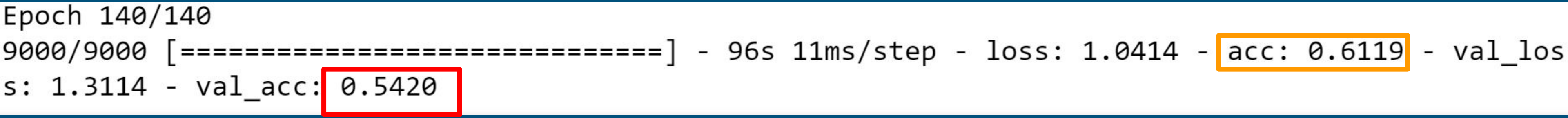
Complex Model

- Added BiDirectional layers to encoders for passage and question
  - BiDirectional LSTMs w/ 64 units
  - Convert to 3D tensor using RepeatVector
  - BiDirectional LSTM w/ 64 units
  - Apply dense layer to each temporal slice of input using TimeDistributed

Code : Reading Comprehension Final Model - with Bidirectional LSTM encoder.ipynb

Complex Model

- The model accuracy with 140 epochs is around 50% accuracy. We tried tweaking various optimization parameters, still the results stayed around 50-54% accuracy for 140 epochs. Interestingly, our simple model beat the complex model in training and test/validation data.



```
Epoch 140/140
9000/9000 [==============================] - 96s 11ms/step - loss: 1.0414 - acc: 0.6119 - val_los
s: 1.3114 - val_acc: 0.5420
```

Notes

- We ran our Simple model code on our own machines in about 10 minutes. No cloud computing needed.
- Complex model with BiDirectional encoders completed and added after presentation in class. Time to run code: 4 hours.