MSc IN DATA SCIENCE
UNIVERSITY OF PELOPONNESE –
NCSR "DEMOKRITOS"

Deep Learning

Assignment 2: Build a Chatbot!

A report by:

Georgios Touros (dsc18023)

January 2020

Data Pre-processing & Vocabulary Creation

We downloaded the original, unprocessed corpus[1] and followed the instructions that were attached in the zip file to pre-process it. Our goal was not to use any of the metadata that are provided, but instead, to simply construct pairs of questions and answers that could be used to train our chatbot.

We started from the “movie_lines.txt” file, keeping only the required columns, namely, "LineID" and "Line". We then stripped the space from "LineID" for further usage and changed the datatype of "Line" to string. We then moved to the “movie_conversations.txt” file, which contained the line sequences. The only necessary field was “conversations”, so we created a list of these conversations, joining them with the actual lines using the LineID column. This initial load takes approximately 5 minutes in our laptop, so we stored the clean “conversations” file as a pickle for further usage.

The resulting file contains too many utterances for our laptop’s RAM to handle, so we decided to take a sample of 30,000 conversations picking random indices from the conversation pickle. Having the new sample of conversations, we then created pairs of questions and answers, by looping through each conversation and extracting a line and its follow-up, ending in the penultimate line. This means that if a conversation included “Hello”, “Greetings. How are you?”, “I am fine, thank you” this would generate two pairs of questions and answers:

- “Hello”, “Greetings. How are you?”
- “Greetings. How are you?”, “I am fine, thank you”

We applied some basic text cleaning and removed all punctuation. Still, the data being too large for our capabilities, we decided to apply 2 measures of sub-sampling:

- Remove pairs of questions and answers when either is larger than a maximum word threshold or smaller than a minimum word threshold. Essentially, we removed from the dataset conversations containing too large or too small lines.
- Remove pairs of questions and answers when either contains words that are too rare in our vocabulary, given a minimum word frequency threshold.

Controlling either of those thresholds allows us to effectively shorten or enlarge the sample and vocabulary sizes, according to our capacity. These parameters played a significant role in the executability of the code, as well as in the results, as larger sample sizes and vocabularies seemed to improve the chatbot’s eloquence.

As a final pre-processing step, we applied <BOS> (Beginning of Sentence) and <EOS> (Ending of Sentence) tagging in the answers, which is an essential step for the input and output of the decoding module of our model, as will be explained later. We created a vocabulary dictionary, using the Keras Tokenizer and used the same tokenizer to convert the questions and answers to padded sequences of tokens.

The Model & Architecture

The implementation of our architecture is done using TensorFlow 1.15.0, using the Keras distribution that is packaged with it. It seemed to be the most easy-to-use and intuitive way to go about this task, as the keras library allows us to have a higher level interface with the model, handling most of the things that happen under the hood, and giving us more time to spend on understanding our architecture.

[1] https://s3.amazonaws.com/pytorch-tutorial-assets/cornell_movie_dialogs_corpus.zip

We chose to use a sequence to sequence model, making use of Long Short-Term Memory Layers as encoding and decoding modules. The input of each of the two modules is passed through an embedding layer, which transforms the padded sequences of word IDs to representations of appropriate dimensionality to be fed to the model. The HIDDEN_DIM parameter controls the latent dimensionality of each of the two LSTM layers (i.e. the number of LSTM "cells"), and, thus, is also needed as a parameter to construct the dimensions of the embedding layers.

We use the default activation functions in both encoder and decoder LSTMs. These are tanh for the output of the layer and hard sigmoid for the recurrent step. We thought it pointless to search for other activation functions for LSTMs, as we research suggests that they work best with this type of model.[2] We set the "return state" of the encoder to true, as we feed the internal states of the encoder's gated units to initialise the decoder's gated units.

After the decoder LSTM layer, we add a fully connected output layer, with a softmax activation. This is necessary in order to return the words with the highest probability. Due to the nature of the output of this layer, we format the target output (against which the model is calculating its errors) in 3 dimensions: sample size, length of the maximum padded decoder output, and the vocabulary size. We can think of this shape as a one-hot-sequence encoding, as each word ID of the padded sequence is transformed to a one-hot-vector. We should also note here, that the output matrix is shifted to the right by one step, meaning that instead of the beginning of the sentence token, the output starts from a zero-pad. The reason for this will be explained later, at the inference stage.

As the output of the model is an array of probabilities, it seemed natural to use categorical cross-entropy as the loss function, given that it corresponds best to the softmax activation of the Dense output layer[3].

We then proceed to train the model using batches of 100 conversations and 45 epochs, using RMSProp as an optimiser. The choice of RMSProp makes sense, as we tend to have the problem of exploding and vanishing gradients in RNNs, therefore RMSProp seems as a fast and obvious choice. We also try the Adam optimiser, to mediocre results. The choice of batches and epochs is arbitrary at this point, and not much thinking is initially put into it, other than execution time and memory load.

Inference Models

In order to decode test sentences, we need to be able to make inferences out of the given test sentences. For this, the following process is followed:

1) Encode the input sentence and retrieve the initial decoder state
2) Run one step of the decoder with this initial state and a "start of sequence" token as target. The output will be the next target character.
3) Append the target character predicted and repeat.
4) End when an End of Sentence token is consumed.

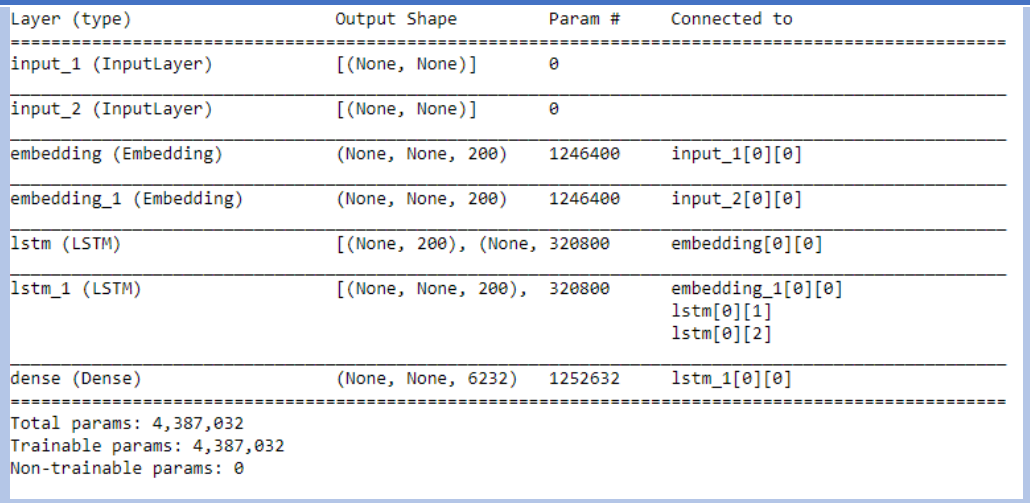
Finally, we make conversations with our Chatbot, using an interactive prompt. The results are summarized in the following section, wherein the clear winner is the 4th model, which is also the one kept in the attached Jupyter Notebook. For convenience and reproducibility purposes, we have also stored and attached the model and weights in json and h5 formats respectively and commented out any piece of code that would run for too long.

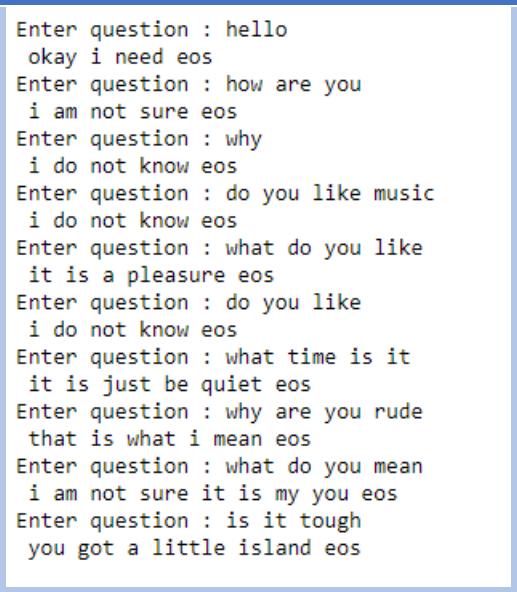
[2] Le, Jaitly, Hinton: A Simple Way to Initialize Recurrent Networks of Rectified Linear Units 2015
[3] Goodfellow, Bengio, Courville: Deep Learning, MIT Press 2016, chapter 5.1

Results

<table>
<tr><th>Parameters</th><th>Model</th><th>Conversations</th></tr>
<tr>
<td>
Min Line Length = 2
Max Line Length = 10
Min Word Frequency = 1
Conversations Sample = 30,000
Questions Used: 13,131
Vocabulary Size: 6232

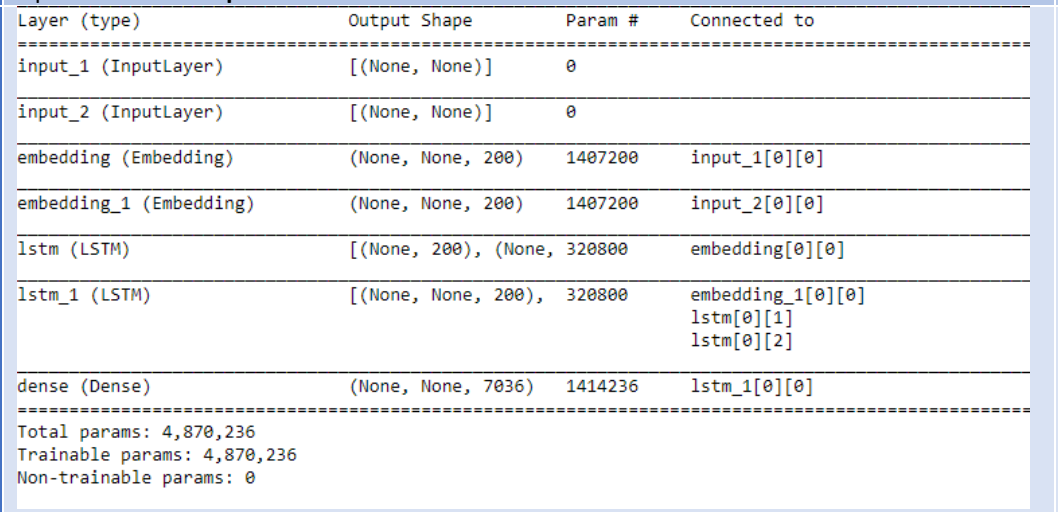
LSTM Hidden Layers = 200
Batch Size = 100
Epochs = 45

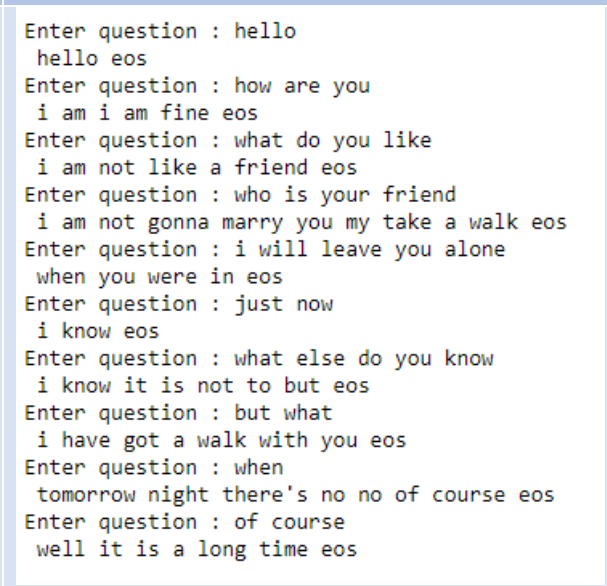
Wall time: 30min 38s
</td>
<td>


LSTM Activation Function: tanh
Recurrent Activation Function: hard_sigmoid
Dense Layer Activation: softmax
Optimiser: RMSProp
</td>
<td>


</td>
</tr>
<tr>
<td>
Min Line Length = 1
Max Line Length = 12
Min Word Frequency = 2
Conversations Sample = 30,000
Questions Used: 27,750
Vocabulary Size: 7036

LSTM Hidden Layers = 200
Batch Size = 100
Epochs = 45

Wall time: 1h 25min 7s
</td>
<td>


LSTM Activation Function: tanh
Recurrent Activation Function: hard_sigmoid
Dense Layer Activation: softmax
Optimiser: RMSProp
</td>
<td>


</td>
</tr>
</table>

Min Line Length = 1
Max Line Length = 12
Min Word Frequency = 2
Conversations Sample = 30,000
Questions Used: 28,229
Vocabulary Size: 7,057

LSTM Hidden Layers = 400
Batch Size = 100
Epochs = 45

Wall time: 2h 41min 58s

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, None)]	0	
input_2 (InputLayer)	[(None, None)]	0	
embedding (Embedding)	(None, None, 400)	2822800	input_1[0][0]
embedding_1 (Embedding)	(None, None, 400)	2822800	input_2[0][0]
lstm (LSTM)	[(None, 400), (None,	1281600	embedding[0][0]
lstm_1 (LSTM)	[(None, None, 400),	1281600	embedding_1[0][0] lstm[0][1] lstm[0][2]
dense (Dense)	(None, None, 7057)	2829857	lstm_1[0][0]

Total params: 11,038,657
Trainable params: 11,038,657
Non-trainable params: 0

LSTM Activation Function: **tanh**
Recurrent Activation Function: **hard_sigmoid**
Dense Layer Activation: **softmax**
Optimiser: Adam

```
Enter question : hello
 it is me eos
Enter question : hi
 hi roses eos
Enter question : who is roses
 i was the creator eos
Enter question : of what
 the hotel eos
Enter question : which hotel
 you will your last did the answer to eos
Enter question : what is your name
 gretchen we are going together all right eos
Enter question : where
 government recently retired eos
Enter question : CIA
 more or never ever gonna help me eos
Enter question : did they
 a little is not really eos
Enter question : what is it then
 i i just do not want to eos
```

Min Line Length = 1
Max Line Length = 12
Min Word Frequency = 2
Conversations Sample = 30,000
Questions Used: 28,229
Vocabulary Size: 7,057

LSTM Hidden Layers = 400
Batch Size = 100
Epochs = 45

Wall time: 2h 35min 58s

Layer (type)	Output Shape	Param #	Connected to
input_13 (InputLayer)	[(None, None)]	0	
input_14 (InputLayer)	[(None, None)]	0	
embedding_4 (Embedding)	(None, None, 400)	2822800	input_13[0][0]
embedding_5 (Embedding)	(None, None, 400)	2822800	input_14[0][0]
lstm_4 (LSTM)	[(None, 400), (None,	1281600	embedding_4[0][0]
lstm_5 (LSTM)	[(None, None, 400),	1281600	embedding_5[0][0] lstm_4[0][1] lstm_4[0][2]
dense_2 (Dense)	(None, None, 7057)	2829857	lstm_5[0][0]

Total params: 11,038,657
Trainable params: 11,038,657
Non-trainable params: 0

LSTM Activation Function: **tanh**
Recurrent Activation Function: **hard_sigmoid**
Dense Layer Activation: **softmax**
Optimiser: RMSProp

```
Enter question : hello
 i am sorry did i wake you eos
Enter question : not at all
 good for you eos
Enter question : how are you
 fine eos
Enter question : what would you like for breakfast
 about what eos
Enter question : breakfast
 yeah eos
Enter question : do you want milk
 sure eos
Enter question : anything else
 what are we going to do about this eos
Enter question : about what
 about where she went i mean what bar eos
Enter question : should we tell the world about it
 well i just got it eos
Enter question : and what will you do
 i am sorry eos
```