Try different features and algorithms

Some of you have already a working model after the previous assignment, and some still have not completed that. For the next deadline, I would like to see some results from your models. So, what I need is really an evaluation of your working model using the evaluation metrics that make sense for your task, this includes accuracy, Precision, Recall, F1, or maybe something specific for your task if these metrics do not work for you.

Some of you implement one algorithm, but for the projects which use tools, evaluate your features try a few different learning algorithms.

Describe what algorithms and via which tools you have used. Evaluate various models and report performance of your model on your test data. You can provide the evaluation of your test data or use cross-validation. Please be communicative about it and let me know what is your approach to improving your initial results. Post your main test results on canvas and codes on your GitHub repository.

Please find my code in Link.

The method applied is a sequence-to-sequence model that consists of couple of algorithms:

1. a stacked LSTM encoder and a stacked LSTM decoder

2. attentions mechanism in on the top of decoder to improve final output by using source LSTM

3. learning rate decay to quickly go to the minimum

4. adam optimizer and clipped gradient to optimize the gradient descent.

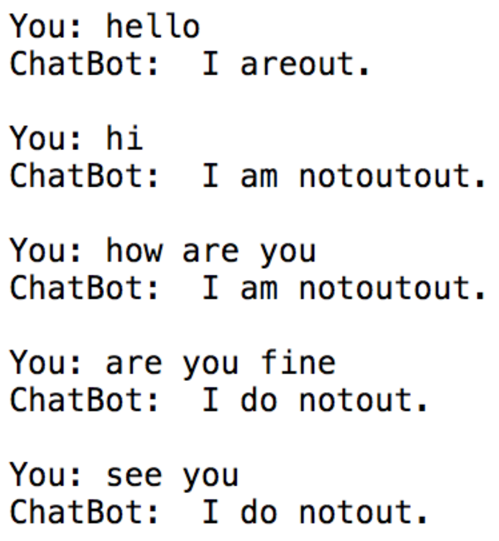
5. dropout method of LSTM to avoid overfitting.

They all can be applied to improve the training weights.

Following are two of my training results.

1.

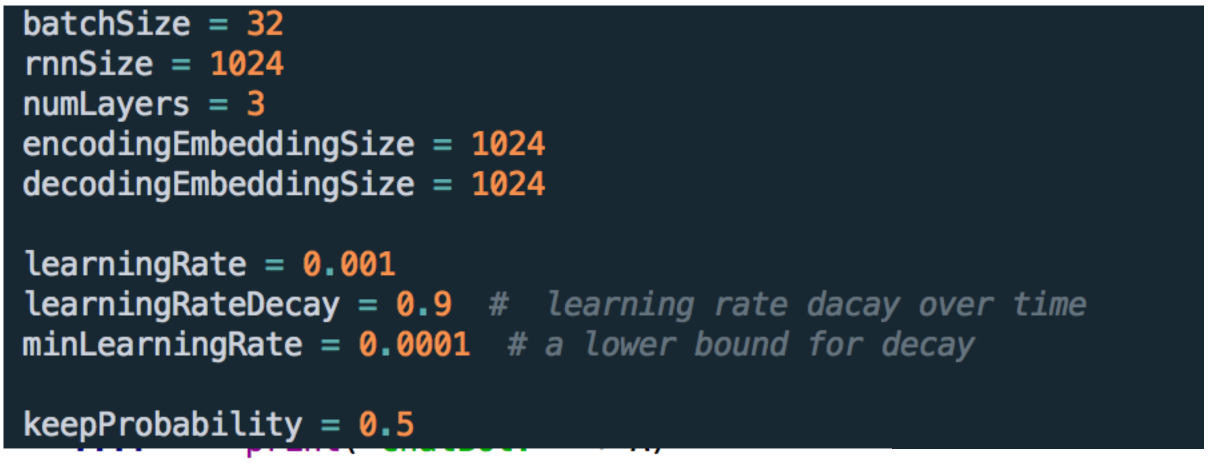
```
                  Epoch:   9/10,
                  Batch:    0/79,
                  Training Loss Error:  1.161,
                  Training Time on 100 Batches: 169 seconds

Validation Loss Error:  1.895,
                  Batch Validation Time: 10 seconds
I speak better now!

                  Epoch:   9/10,
                  Batch:   50/79,
                  Training Loss Error:  1.924,
                  Training Time on 100 Batches: 199 seconds

Validation Loss Error:  1.903,
                  Batch Validation Time: 10 seconds
Sorry I do not speak better, I need to practice more.

                  Epoch:  10/10,
                  Batch:    0/79,
                  Training Loss Error:  1.154,
                  Training Time on 100 Batches: 172 seconds

Validation Loss Error:  1.891,
                  Batch Validation Time: 10 seconds
I speak better now!

                  Epoch:  10/10,
                  Batch:   50/79,
                  Training Loss Error:  1.912,
                  Training Time on 100 Batches: 201 seconds

Validation Loss Error:  1.887,
                  Batch Validation Time: 10 seconds
I speak better now!
Game Over
```



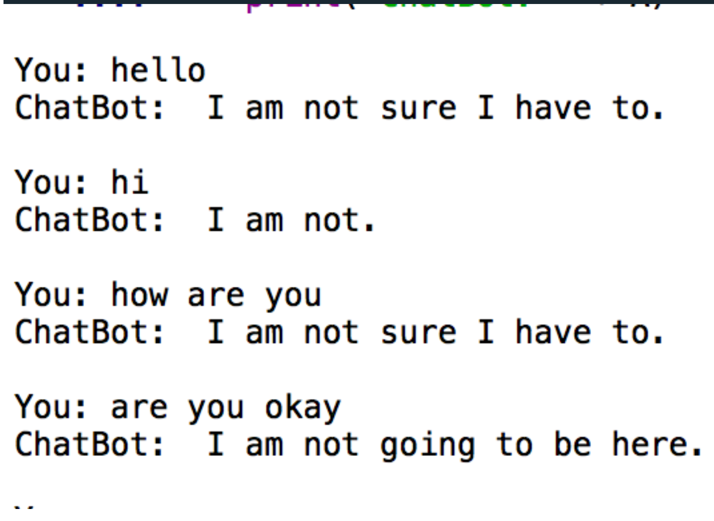
```
You: hello
ChatBot:  I areout.

You: hi
ChatBot:  I am notoutout.

You: how are you
ChatBot:  I am notoutout.

You: are you fine
ChatBot:  I do notout.

You: see you
ChatBot:  I do notout.
```

I just run 10 epochs on 1% of the original dataset. The results are very simple. And it will return a lot of “out”, since the vocabulary is small. (A lot of words are not covered.)

2.



```
batchSize = 32
rnnSize = 1024
numLayers = 3
encodingEmbeddingSize = 1024
decodingEmbeddingSize = 1024

learningRate = 0.001
learningRateDecay = 0.9  #  learning rate dacay over time
minLearningRate = 0.0001  # a lower bound for decay

keepProbability = 0.5
```



```
You: hello
ChatBot:  I am not sure I have to.

You: hi
ChatBot:  I am not.

You: how are you
ChatBot:  I am not sure I have to.

You: are you okay
ChatBot:  I am not going to be here.
```

This result is on the whole dataset but only the run 2 epochs, with the hyperparameters above. It can be seen the result is somehow more reasonable.

I am still trying to get more powerful GPU resource. I hope I can get much more meaningful result.