CODE SUMMARY

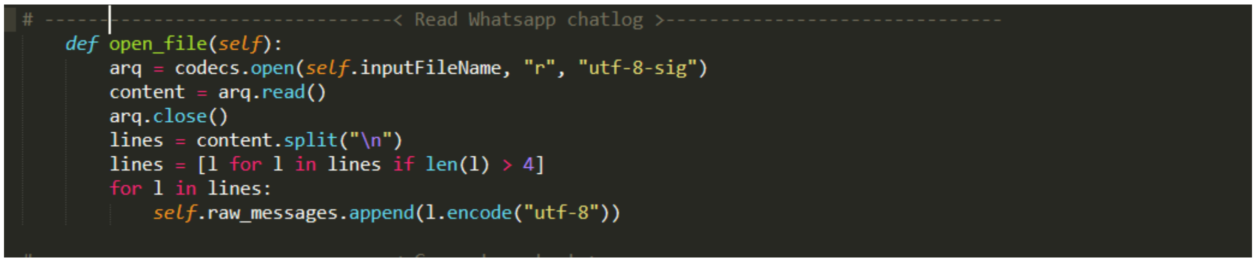
For this course's final project, I am building a Chatbot. Here is a summary of my results. Here are the file names and their objectives:

- Parser_1: Extracts data from What's-app log and creates a csv
- Process_whatsapp_data_2: Builds padded input sequences (question, answer)
- Train_bot_3: training takes place in this file
- Test_bot_4: testing takes place in this file

Dataset used: Movie dialogue and WhatsApp log

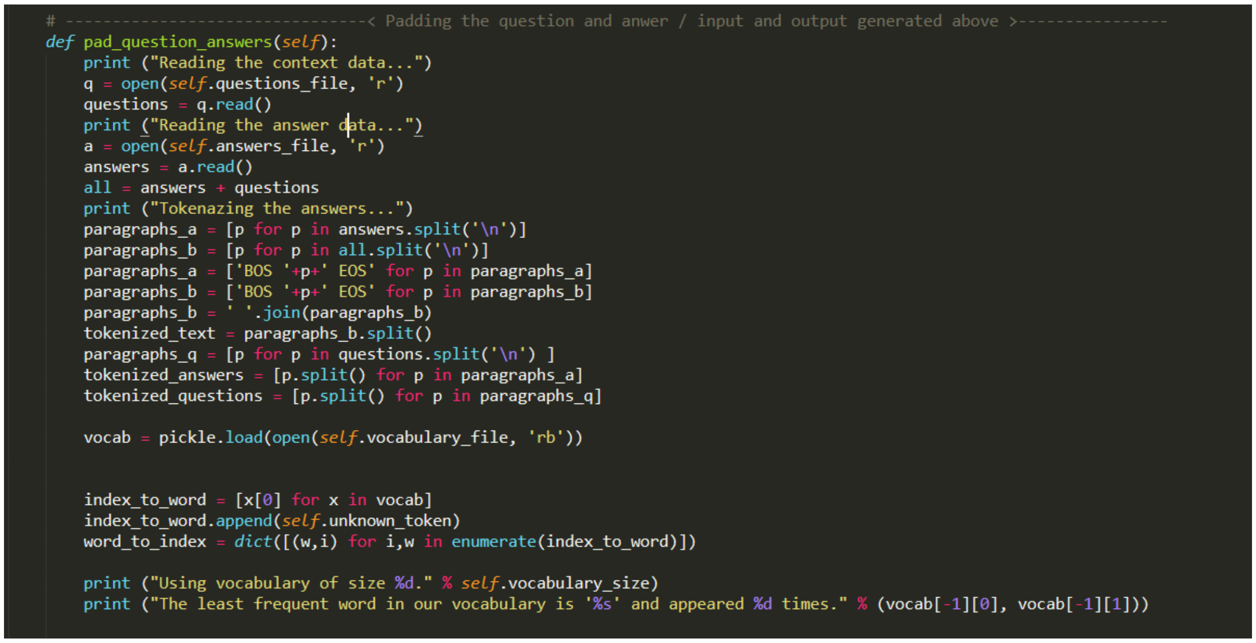
To view text files in formatted space use Sublime.txt or Notepad++. I am still trying to improve the context of the bot replies.

EXPLANATION: Here I am extracting WhatsApp conversation logs from my WhatsApp chat history. WhatsApp has a feature that allows us to view all chat history in text format. After processing this data, I trained my model on this data.



```
# ---------------------------------< Read Whatsapp chatlog >------------------------------
    def open_file(self):
        arq = codecs.open(self.inputFileName, "r", "utf-8-sig")
        content = arq.read()
        arq.close()
        lines = content.split("\n")
        lines = [l for l in lines if len(l) > 4]
        for l in lines:
            self.raw_messages.append(l.encode("utf-8"))
```

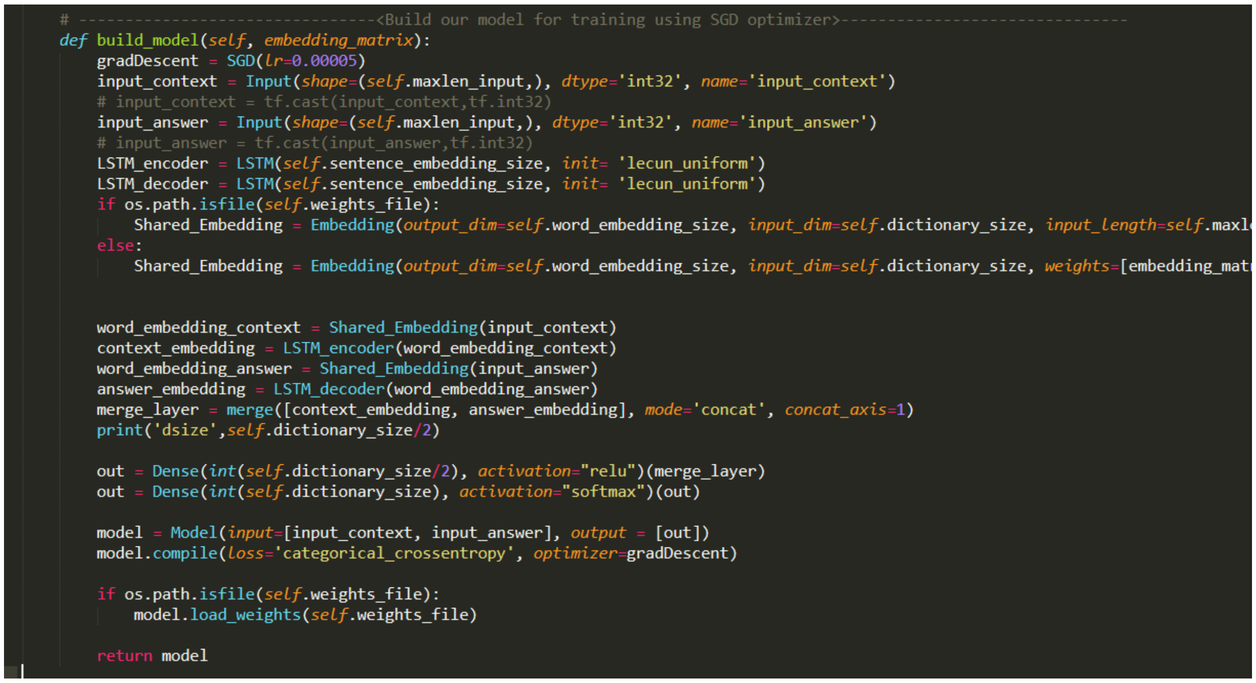
After obtaining our dataset we perform padding which pre-fixes our input sequence length.



```
# --------------------------------< Padding the question and anwer / input and output generated above >----------------
def pad_question_answers(self):
    print ("Reading the context data...")
    q = open(self.questions_file, 'r')
    questions = q.read()
    print ("Reading the answer data...")
    a = open(self.answers_file, 'r')
    answers = a.read()
    all = answers + questions
    print ("Tokenazing the answers...")
    paragraphs_a = [p for p in answers.split('\n')]
    paragraphs_b = [p for p in all.split('\n')]
    paragraphs_a = ['BOS '+p+' EOS' for p in paragraphs_a]
    paragraphs_b = ['BOS '+p+' EOS' for p in paragraphs_b]
    paragraphs_b = ' '.join(paragraphs_b)
    tokenized_text = paragraphs_b.split()
    paragraphs_q = [p for p in questions.split('\n') ]
    tokenized_answers = [p.split() for p in paragraphs_a]
    tokenized_questions = [p.split() for p in paragraphs_q]

    vocab = pickle.load(open(self.vocabulary_file, 'rb'))


    index_to_word = [x[0] for x in vocab]
    index_to_word.append(self.unknown_token)
    word_to_index = dict([(w,i) for i,w in enumerate(index_to_word)])

    print ("Using vocabulary of size %d." % self.vocabulary_size)
    print ("The least frequent word in our vocabulary is '%s' and appeared %d times." % (vocab[-1][0], vocab[-1][1]))
```

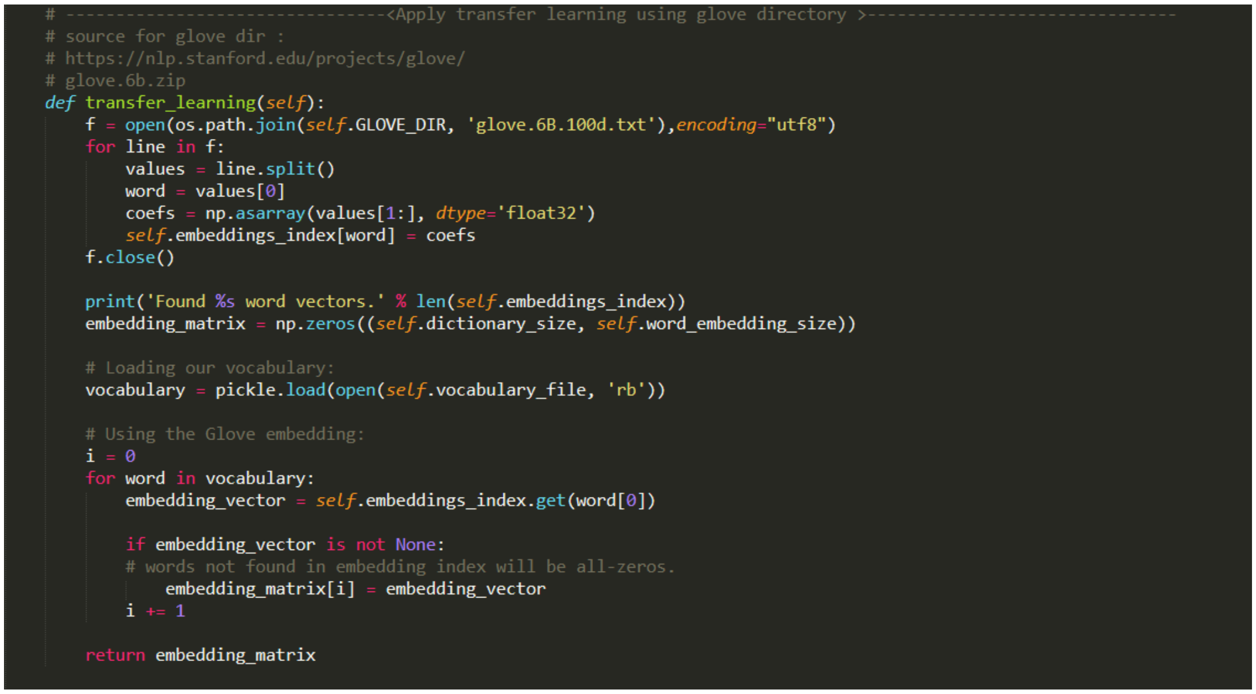
OUR MODEL:

This image demonstrates how easy it is to design our model using keras and then perform optimizations on it. I have used SGD (Stochastic gradient descent) as the optimizer.



```
# -------------------------------<Build our model for training using SGD optimizer>------------------------------
def build_model(self, embedding_matrix):
    gradDescent = SGD(lr=0.00005)
    input_context = Input(shape=(self.maxlen_input,), dtype='int32', name='input_context')
    # input_context = tf.cast(input_context,tf.int32)
    input_answer = Input(shape=(self.maxlen_input,), dtype='int32', name='input_answer')
    # input_answer = tf.cast(input_answer,tf.int32)
    LSTM_encoder = LSTM(self.sentence_embedding_size, init= 'lecun_uniform')
    LSTM_decoder = LSTM(self.sentence_embedding_size, init= 'lecun_uniform')
    if os.path.isfile(self.weights_file):
        Shared_Embedding = Embedding(output_dim=self.word_embedding_size, input_dim=self.dictionary_size, input_length=self.maxl
    else:
        Shared_Embedding = Embedding(output_dim=self.word_embedding_size, input_dim=self.dictionary_size, weights=[embedding_mat


    word_embedding_context = Shared_Embedding(input_context)
    context_embedding = LSTM_encoder(word_embedding_context)
    word_embedding_answer = Shared_Embedding(input_answer)
    answer_embedding = LSTM_decoder(word_embedding_answer)
    merge_layer = merge([context_embedding, answer_embedding], mode='concat', concat_axis=1)
    print('dsize',self.dictionary_size/2)

    out = Dense(int(self.dictionary_size/2), activation="relu")(merge_layer)
    out = Dense(int(self.dictionary_size), activation="softmax")(out)

    model = Model(input=[input_context, input_answer], output = [out])
    model.compile(loss='categorical_crossentropy', optimizer=gradDescent)

    if os.path.isfile(self.weights_file):
        model.load_weights(self.weights_file)

    return model
```

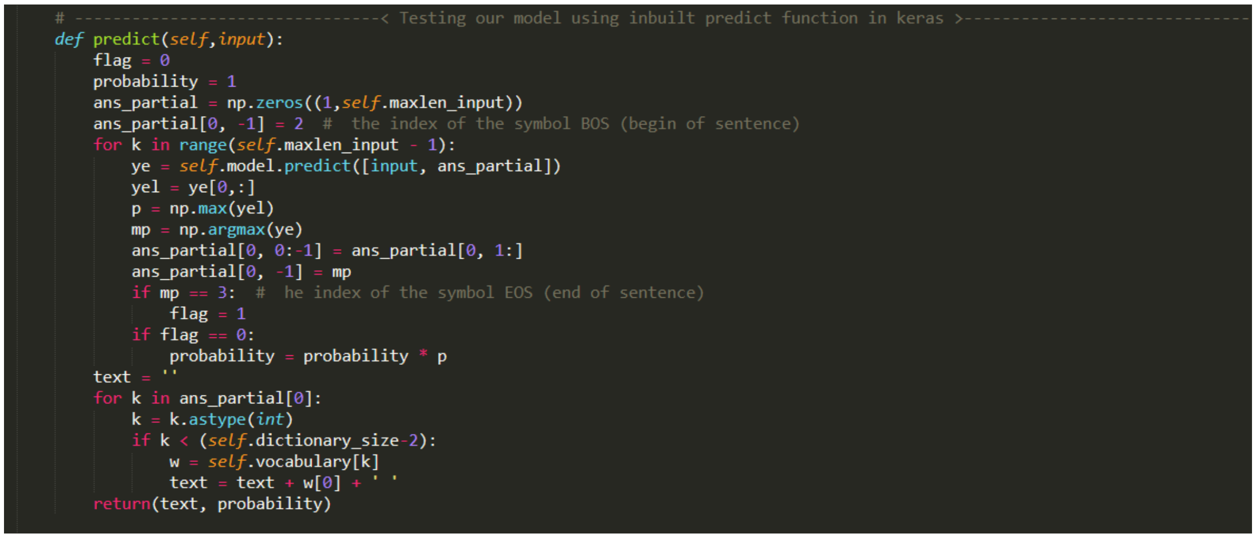
Here I am making use of a pre-trained model to lower my computational effort and results. These are nothing but weights which are embedded and then used in my own model instead of using random values.



```
# --------------------------------<Apply transfer learning using glove directory >------------------------------
# source for glove dir :
# https://nlp.stanford.edu/projects/glove/
# glove.6b.zip
def transfer_learning(self):
    f = open(os.path.join(self.GLOVE_DIR, 'glove.6B.100d.txt'),encoding="utf8")
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        self.embeddings_index[word] = coefs
    f.close()

    print('Found %s word vectors.' % len(self.embeddings_index))
    embedding_matrix = np.zeros((self.dictionary_size, self.word_embedding_size))

    # Loading our vocabulary:
    vocabulary = pickle.load(open(self.vocabulary_file, 'rb'))

    # Using the Glove embedding:
    i = 0
    for word in vocabulary:
        embedding_vector = self.embeddings_index.get(word[0])

        if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
            embedding_matrix[i] = embedding_vector
        i += 1

    return embedding_matrix
```

RESULTS:

This is how we test our model by using the model.predict inbuilt function available in keras. The arguments include the input parameters. Most probable result is printed, and probability is also displayed as part of result.

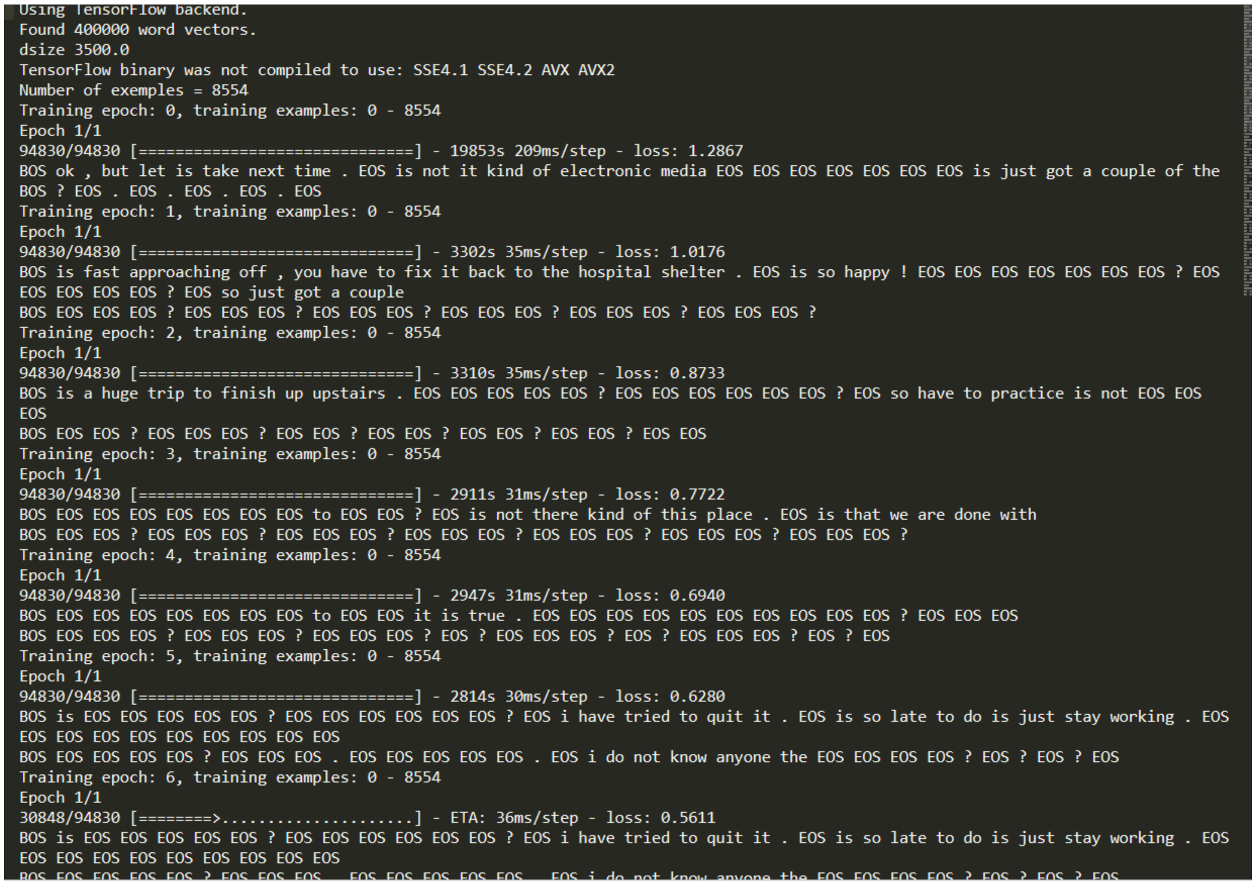


```
# --------------------------------< Testing our model using inbuilt predict function in keras >-----------------------------
def predict(self,input):
    flag = 0
    probability = 1
    ans_partial = np.zeros((1,self.maxlen_input))
    ans_partial[0, -1] = 2  #  the index of the symbol BOS (begin of sentence)
    for k in range(self.maxlen_input - 1):
        ye = self.model.predict([input, ans_partial])
        yel = ye[0,:]
        p = np.max(yel)
        mp = np.argmax(ye)
        ans_partial[0, 0:-1] = ans_partial[0, 1:]
        ans_partial[0, -1] = mp
        if mp == 3:  #  he index of the symbol EOS (end of sentence)
            flag = 1
        if flag == 0:
            probability = probability * p
    text = ''
    for k in ans_partial[0]:
        k = k.astype(int)
        if k < (self.dictionary_size-2):
            w = self.vocabulary[k]
            text = text + w[0] + ' '
    return(text, probability)
```

RESULT ANALYSIS:

Train log: Full log can be found in the Traininglog.txt file

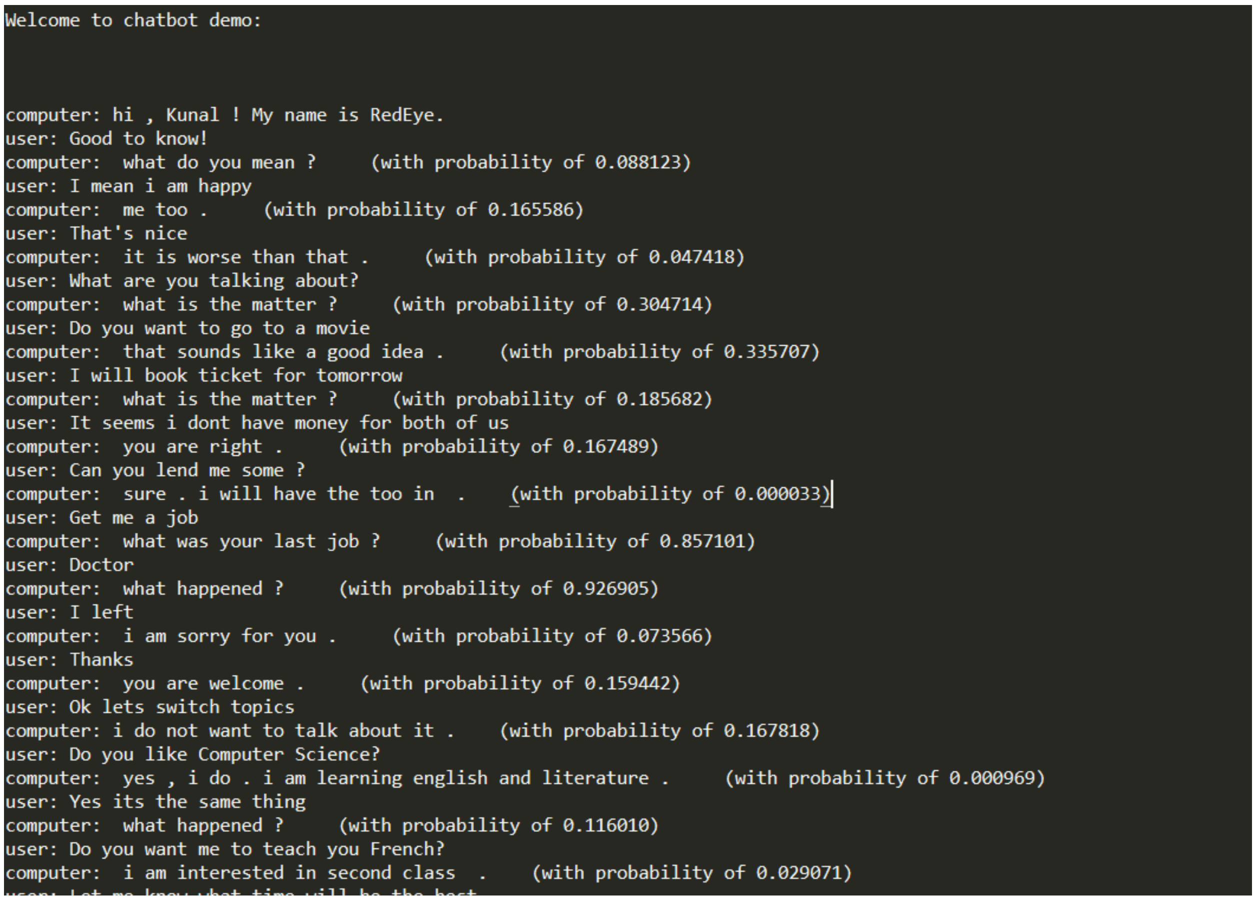
As we can see the model is training and the error value is decreasing. The text below each epoch is one instance of training input



```
Using TensorFlow backend.
Found 400000 word vectors.
dsize 3500.0
TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2
Number of exemples = 8554
Training epoch: 0, training examples: 0 - 8554
Epoch 1/1
94830/94830 [==============================] - 19853s 209ms/step - loss: 1.2867
BOS ok , but let is take next time . EOS is not it kind of electronic media EOS EOS EOS EOS EOS EOS EOS is just got a couple of the
BOS ? EOS . EOS . EOS . EOS . EOS
Training epoch: 1, training examples: 0 - 8554
Epoch 1/1
94830/94830 [==============================] - 3302s 35ms/step - loss: 1.0176
BOS is fast approaching off , you have to fix it back to the hospital shelter . EOS is so happy ! EOS EOS EOS EOS EOS EOS EOS ? EOS
EOS EOS EOS EOS ? EOS so just got a couple
BOS EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ?
Training epoch: 2, training examples: 0 - 8554
Epoch 1/1
94830/94830 [==============================] - 3310s 35ms/step - loss: 0.8733
BOS is a huge trip to finish up upstairs . EOS EOS EOS EOS EOS ? EOS EOS EOS EOS EOS EOS ? EOS so have to practice is not EOS EOS
EOS
BOS EOS EOS ? EOS EOS EOS ? EOS EOS ? EOS EOS ? EOS EOS ? EOS EOS ? EOS EOS
Training epoch: 3, training examples: 0 - 8554
Epoch 1/1
94830/94830 [==============================] - 2911s 31ms/step - loss: 0.7722
BOS EOS EOS EOS EOS EOS EOS EOS to EOS EOS ? EOS is not there kind of this place . EOS is that we are done with
BOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ?
Training epoch: 4, training examples: 0 - 8554
Epoch 1/1
94830/94830 [==============================] - 2947s 31ms/step - loss: 0.6940
BOS EOS EOS EOS EOS EOS EOS EOS to EOS EOS it is true . EOS EOS EOS EOS EOS EOS EOS EOS EOS EOS ? EOS EOS EOS
BOS EOS EOS EOS ? EOS EOS EOS ? EOS EOS EOS ? EOS ? EOS EOS EOS ? EOS ? EOS EOS EOS ? EOS ? EOS
Training epoch: 5, training examples: 0 - 8554
Epoch 1/1
94830/94830 [==============================] - 2814s 30ms/step - loss: 0.6280
BOS is EOS EOS EOS EOS EOS ? EOS EOS EOS EOS EOS EOS ? EOS i have tried to quit it . EOS is so late to do is just stay working . EOS
EOS EOS EOS EOS EOS EOS EOS EOS EOS
BOS EOS EOS EOS EOS ? EOS EOS EOS . EOS EOS EOS EOS EOS . EOS i do not know anyone the EOS EOS EOS EOS ? EOS ? EOS ? EOS
Training epoch: 6, training examples: 0 - 8554
Epoch 1/1
30848/94830 [=======>......................] - ETA: 36ms/step - loss: 0.5611
BOS is EOS EOS EOS EOS EOS ? EOS EOS EOS EOS EOS EOS ? EOS i have tried to quit it . EOS is so late to do is just stay working . EOS
EOS EOS EOS EOS EOS EOS EOS EOS EOS
BOS EOS EOS EOS EOS ? EOS EOS EOS . EOS EOS EOS EOS EOS . EOS i do not know anyone the EOS EOS EOS EOS ? EOS ? EOS ? EOS
```

Test log result analysis:

- Full log can be found in the result.txt file.
- As we can see the bot does make sense and is some cases out of context.
- Training time was long since I did not use a GPU. The total time for each epoch took less than 20 - 25 minutes and after 18 epochs the error value did not decrease so I stopped.
- I am still trying to train my model on other datasets and plan to pre-process data more to improve context in my results.



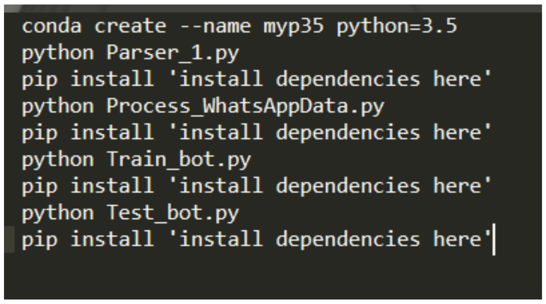
```
Welcome to chatbot demo:



computer: hi , Kunal ! My name is RedEye.
user: Good to know!
computer:  what do you mean ?     (with probability of 0.088123)
user: I mean i am happy
computer:  me too .     (with probability of 0.165586)
user: That's nice
computer:  it is worse than that .     (with probability of 0.047418)
user: What are you talking about?
computer:  what is the matter ?     (with probability of 0.304714)
user: Do you want to go to a movie
computer:  that sounds like a good idea .     (with probability of 0.335707)
user: I will book ticket for tomorrow
computer:  what is the matter ?     (with probability of 0.185682)
user: It seems i dont have money for both of us
computer:  you are right .     (with probability of 0.167489)
user: Can you lend me some ?
computer:  sure . i will have the too in  .    (with probability of 0.000033)
user: Get me a job
computer:  what was your last job ?     (with probability of 0.857101)
user: Doctor
computer:  what happened ?     (with probability of 0.926905)
user: I left
computer:  i am sorry for you .     (with probability of 0.073566)
user: Thanks
computer:  you are welcome .     (with probability of 0.159442)
user: Ok lets switch topics
computer: i do not want to talk about it .    (with probability of 0.167818)
user: Do you like Computer Science?
computer:  yes , i do . i am learning english and literature .     (with probability of 0.000969)
user: Yes its the same thing
computer:  what happened ?     (with probability of 0.116010)
user: Do you want me to teach you French?
computer:  i am interested in second class  .    (with probability of 0.029071)
```

BUILD INSTRUCTIONS:

First install Anaconda to set up the build environment.

I used python version 3.5 since tensor flow support was not there for certain libraries. Here are the build commands for Linux.

Required files: Vocabulary_file.txt, data.txt (Movie Data), Traininglog.txt (WhatsApp data), Glove6B (Transfer learning)



```
conda create --name myp35 python=3.5
python Parser_1.py
pip install 'install dependencies here'
python Process_WhatsAppData.py
pip install 'install dependencies here'
python Train_bot.py
pip install 'install dependencies here'
python Test_bot.py
pip install 'install dependencies here'
```