Generative_Chatbot_LSTM+Attention_TensorFlow

January 14, 2019

1 Generative Chatbot using RNNs (LSTM) & Attention in TensorFlow

1.1 Purpose

The aim of this project is **to make a generative chatbot as your digital avatar.** Your speech consists of your personal voice and words. State of the art Deep Learning techniques, viz. **Sequence to Sequence modelling and Attention models** are widely used to 1. Clone personal voice 2. Replicate talking style and language

Voice cloning efforts such as **Samsung Bixby** which aims to preserve voice of our loved ones, or **Baidu's 'Deep Voice' AI System** addresses first half of the problem. This project focus on the latter half, i.e. to make a personified text-based chatbot, as your digital avatar.

As our aim is to make a more human-like system, we would choose to make the more powerful Generative model.

1.2 At a glance

We have **used Recurrent Neural Networks (LSTMs),** that is the go-to architecture to solve Seq2Seq problems coupled **with Attention mechanism make a generative chatbot. The model is trained using personal chat conversations from Whatsapp and Telegram.**

All the conversation datasets are parsed and converted to the same format to feed seq2seq model. Both participants are marked with 2 symbols at the starting of each line. The parser code process all the files inside "DATA_DIR_PATH" folder.

Forward and Reverse mapping dictionaries for Word2Index and Index2Word conversion is created. Input sequence (words) are converted to indices using Word2Index and are padded to same length, for batch input to encoder. Output from encoder are converted from integer to words using Index2Word mapping.

To train the model, **the padded input and output sequences (indices) from the above step are fed to the S2S architecture.** The embedding layer convert words to indices, which are fed to multiple LSTM cells stacked together in hidden layers.

Interestingly, the chat-bot is found to give responses similar in style to the personal data used for training. Still there are a few grammatical errors, typical of generative models. But as we add more and more training data & tune the hyper-parameters to minimize the loss value, the bot behaviour is found increasingly stable.

1.2.1 Datasets

I have found that training with **only real-word chat conversations between 2 humans doesn't produce stable results.** Chat messages usually contains acronyms (like 'brb', 'lol' etc), shorthand

, net slang and typos to confuse neural network training. Hence, I have **used a combination of real-world chat messages and human-bot interactions to train.**

1) Personal chat **conversations from Whatsapp and Telegram** (downloaded as HTML files) {only parser included}
2) The **Human-Bot Conversational Intelligence Challenge 2 (ConvAI2) conversational dataset** conducted under the scope of NIPS (NeurIPS) 2018 Competition (JSON files) http://convai.io/data/
3) Conversational **GuntherCox Dataset (English)** obtained from: https://github.com/gunthercox/chatterbot-corpus/tree/master/chatterbot_corpus/data/english
4) **Human Conversations** obtained from: https://www.kaggle.com/eibriel/rdany-conversations

1.3 Dependencies & Initializations

```
In [1]: import matplotlib.pyplot as plt
        import tensorflow as tf
        from sklearn.model_selection import train_test_split
        import numpy as np

        from collections import Counter
        import os
        import nltk
```

```
In [2]: alphas = 'abcdefghijklmnopqrstuvwxyz1234567890'

        MAX_INPUT_SEQ_LENGTH = 40
        MAX_TARGET_SEQ_LENGTH = 40
        DATA_DIR_PATH = 'data'
        MAX_VOCAB_SIZE = 30000

        marker_start = '<begin>'
        marker_end = '<end>'
        marker_unknown = '<unk>'
        marker_pad = '<pad>'

        # standard step - reset computation graphs
        tf.reset_default_graph()

        # 2 more for start and stop markers
        input_seq_len = 15
        output_seq_len = input_seq_len+2
```

```
In [3]: #defines permissible characters for the chatbot
        def permissible_chars(word):

            for char in word:
```

```
            if char in alphas:
                return True

        return False
```

```
In [4]: # Compute softmax values for each sets of scores in x.
        def softmax(x):
            e_x = np.exp(x - np.max(x))
            return e_x / e_x.sum()
```

1.4 Data Parser & Conversion

To take formatted chat conversations in *.yml files inside ./DATA_DIR_PATH and convert each common words in each line indices.

```
In [5]: # To parse the input yml files and create word2index and index2word mappings

        target_counter = Counter()
        input_counter = Counter()

        input_texts = []
        target_texts = []

        # Parser base code of GuntherCox dataset obtained from link below and modified as per
        # https://github.com/kushagra2101/ChatCrazie/blob/master/train_seq2seq.py

        for file in os.listdir(DATA_DIR_PATH):
            filepath = os.path.join(DATA_DIR_PATH, file)
            if os.path.isfile(filepath):
                print('processing file: ', file)
                lines = open(filepath, 'rt', encoding='utf8').read().split('\n')
                prev_words = []
                for line in lines:

                    if line.startswith('- - '):
                        prev_words = []

                    if line.startswith('- - ') or line.startswith('  - '):
                        line = line.replace('- - ', '')
                        line = line.replace('  - ', '')
                        next_words = [w.lower() for w in nltk.word_tokenize(line)]
                        next_words = [w for w in next_words if permissible_chars(w)]
                        if len(next_words) > MAX_TARGET_SEQ_LENGTH:
                            next_words = next_words[0:MAX_TARGET_SEQ_LENGTH]

                        if len(prev_words) > 0:
                            input_texts.append(prev_words)
                            for w in prev_words:
```

```
                    input_counter[w] += 1

                target_words = next_words[:]
                for w in target_words:
                    target_counter[w] += 1
                target_texts.append(target_words)

            prev_words = next_words


for idx, (input_words, target_words) in enumerate(zip(input_texts, target_texts)):
    if idx < 20:
        print([input_words, target_words])

input_w2i, input_i2w, target_w2i, target_i2w = {},{},{},{}

### Creating Word2index and Index2word, forward and reverse mapping ###
## we will create dictionaries to provide a unique integer for each word.
input_w2i[marker_unknown] = 0
input_w2i[marker_pad] = 1
# filter out the rare words
for idx, word in enumerate(input_counter.most_common(MAX_VOCAB_SIZE)):
    input_w2i[word[0]] = idx+2

# inverse dictionary for vocab_to_int.
input_i2w = dict([(idx, word) for word, idx in input_w2i.items()])

## we will create dictionaries to provide a unique integer for each word.
target_w2i[marker_unknown] = 0
target_w2i[marker_pad] = 1
target_w2i[marker_start] = 2
target_w2i[marker_end] = 3
for idx, word in enumerate(target_counter.most_common(MAX_VOCAB_SIZE)):
    target_w2i[word[0]] = idx+4

# inverse dictionary for vocab_to_int.
target_i2w = dict([(idx, word) for word, idx in target_w2i.items()])


##############################################

# inputVocabLen = len(input_word2idx)
# targetVocabLen = len(target_word2idx)

##############################################

# if the word is not found then default with 0.
# 0 in index means the word is unknown (<unk>)
```

```
        x = [[input_w2i.get(word, 0) for word in sentence] for sentence in input_texts]
        y = [[target_w2i.get(word, 0) for word in sentence] for sentence in target_texts]

        inputVocabLen = len(input_w2i)
        targetVocabLen = len(target_w2i)
```

```
processing file:  Artificial_intelligence.yml
processing file:  bot_info.yml
processing file:  computers.yml
processing file:  conversations.yml
processing file:  danny.yml
processing file:  data_intermediate.yml
processing file:  data_tolokers.yml
processing file:  data_volunteers.yml
processing file:  emotion.yml
processing file:  film.yml
processing file:  food.yml
processing file:  general convo.yml
processing file:  gentyped.txt
processing file:  GK.yml
processing file:  gossip.yml
processing file:  greetings.yml
processing file:  health.yml
processing file:  IT.yml
processing file:  jokes_fun.yml
processing file:  market_money.yml
processing file:  messages.html.txt
processing file:  messages10.html.txt
processing file:  messages100.html.txt
processing file:  messages101.html.txt
processing file:  messages102.html.txt
processing file:  messages103.html.txt
processing file:  messages104.html.txt
processing file:  messages105.html.txt
processing file:  messages106.html.txt
processing file:  messages107.html.txt
processing file:  messages108.html.txt
processing file:  messages109.html.txt
processing file:  messages11.html.txt
processing file:  messages110.html.txt
processing file:  messages111.html.txt
processing file:  messages112.html.txt
processing file:  messages113.html.txt
processing file:  messages114.html.txt
processing file:  messages115.html.txt
processing file:  messages116.html.txt
processing file:  messages117.html.txt
processing file:  messages118.html.txt
```

```
processing file:  messages119.html.txt
processing file:  messages12.html.txt
processing file:  messages120.html.txt
processing file:  messages121.html.txt
processing file:  messages122.html.txt
processing file:  messages123.html.txt
processing file:  messages124.html.txt
processing file:  messages125.html.txt
processing file:  messages126.html.txt
processing file:  messages127.html.txt
processing file:  messages128.html.txt
processing file:  messages129.html.txt
processing file:  messages13.html.txt
processing file:  messages130.html.txt
processing file:  messages131.html.txt
processing file:  messages132.html.txt
processing file:  messages133.html.txt
processing file:  messages134.html.txt
processing file:  messages135.html.txt
processing file:  messages136.html.txt
processing file:  messages137.html.txt
processing file:  messages138.html.txt
processing file:  messages139.html.txt
processing file:  messages14.html.txt
processing file:  messages140.html.txt
processing file:  messages141.html.txt
processing file:  messages142.html.txt
processing file:  messages143.html.txt
processing file:  messages144.html.txt
processing file:  messages145.html.txt
processing file:  messages146.html.txt
processing file:  messages147.html.txt
processing file:  messages148.html.txt
processing file:  messages149.html.txt
processing file:  messages15.html.txt
processing file:  messages150.html.txt
processing file:  messages151.html.txt
processing file:  messages152.html.txt
processing file:  messages153.html.txt
processing file:  messages154.html.txt
processing file:  messages155.html.txt
processing file:  messages156.html.txt
processing file:  messages157.html.txt
processing file:  messages158.html.txt
processing file:  messages159.html.txt
processing file:  messages16.html.txt
processing file:  messages160.html.txt
processing file:  messages161.html.txt
```

```
processing file:  messages162.html.txt
processing file:  messages163.html.txt
processing file:  messages164.html.txt
processing file:  messages165.html.txt
processing file:  messages166.html.txt
processing file:  messages167.html.txt
processing file:  messages168.html.txt
processing file:  messages169.html.txt
processing file:  messages17.html.txt
processing file:  messages170.html.txt
processing file:  messages171.html.txt
processing file:  messages172.html.txt
processing file:  messages173.html.txt
processing file:  messages174.html.txt
processing file:  messages175.html.txt
processing file:  messages176.html.txt
processing file:  messages177.html.txt
processing file:  messages178.html.txt
processing file:  messages179.html.txt
processing file:  messages18.html.txt
processing file:  messages180.html.txt
processing file:  messages181.html.txt
processing file:  messages182.html.txt
processing file:  messages183.html.txt
processing file:  messages184.html.txt
processing file:  messages185.html.txt
processing file:  messages186.html.txt
processing file:  messages187.html.txt
processing file:  messages188.html.txt
processing file:  messages189.html.txt
processing file:  messages19.html.txt
processing file:  messages190.html.txt
processing file:  messages191.html.txt
processing file:  messages192.html.txt
processing file:  messages193.html.txt
processing file:  messages194.html.txt
processing file:  messages195.html.txt
processing file:  messages196.html.txt
processing file:  messages197.html.txt
processing file:  messages198.html.txt
processing file:  messages199.html.txt
processing file:  messages2.html.txt
processing file:  messages20.html.txt
processing file:  messages200.html.txt
processing file:  messages201.html.txt
processing file:  messages202.html.txt
processing file:  messages203.html.txt
processing file:  messages204.html.txt
```

```
processing file:  messages205.html.txt
processing file:  messages206.html.txt
processing file:  messages207.html.txt
processing file:  messages208.html.txt
processing file:  messages209.html.txt
processing file:  messages21.html.txt
processing file:  messages210.html.txt
processing file:  messages211.html.txt
processing file:  messages212.html.txt
processing file:  messages213.html.txt
processing file:  messages214.html.txt
processing file:  messages215.html.txt
processing file:  messages216.html.txt
processing file:  messages217.html.txt
processing file:  messages218.html.txt
processing file:  messages219.html.txt
processing file:  messages22.html.txt
processing file:  messages220.html.txt
processing file:  messages221.html.txt
processing file:  messages222.html.txt
processing file:  messages223.html.txt
processing file:  messages224.html.txt
processing file:  messages225.html.txt
processing file:  messages226.html.txt
processing file:  messages227.html.txt
processing file:  messages228.html.txt
processing file:  messages23.html.txt
processing file:  messages24.html.txt
processing file:  messages25.html.txt
processing file:  messages26.html.txt
processing file:  messages27.html.txt
processing file:  messages28.html.txt
processing file:  messages29.html.txt
processing file:  messages3.html.txt
processing file:  messages30.html.txt
processing file:  messages31.html.txt
processing file:  messages32.html.txt
processing file:  messages33.html.txt
processing file:  messages34.html.txt
processing file:  messages35.html.txt
processing file:  messages36.html.txt
processing file:  messages37.html.txt
processing file:  messages38.html.txt
processing file:  messages39.html.txt
processing file:  messages4.html.txt
processing file:  messages40.html.txt
processing file:  messages41.html.txt
processing file:  messages42.html.txt
```

```
processing file:  messages43.html.txt
processing file:  messages44.html.txt
processing file:  messages45.html.txt
processing file:  messages46.html.txt
processing file:  messages47.html.txt
processing file:  messages48.html.txt
processing file:  messages49.html.txt
processing file:  messages5.html.txt
processing file:  messages50.html.txt
processing file:  messages51.html.txt
processing file:  messages52.html.txt
processing file:  messages53.html.txt
processing file:  messages54.html.txt
processing file:  messages55.html.txt
processing file:  messages56.html.txt
processing file:  messages57.html.txt
processing file:  messages58.html.txt
processing file:  messages59.html.txt
processing file:  messages6.html.txt
processing file:  messages60.html.txt
processing file:  messages61.html.txt
processing file:  messages62.html.txt
processing file:  messages63.html.txt
processing file:  messages64.html.txt
processing file:  messages65.html.txt
processing file:  messages66.html.txt
processing file:  messages67.html.txt
processing file:  messages68.html.txt
processing file:  messages69.html.txt
processing file:  messages7.html.txt
processing file:  messages70.html.txt
processing file:  messages71.html.txt
processing file:  messages72.html.txt
processing file:  messages73.html.txt
processing file:  messages74.html.txt
processing file:  messages75.html.txt
processing file:  messages76.html.txt
processing file:  messages77.html.txt
processing file:  messages78.html.txt
processing file:  messages79.html.txt
processing file:  messages8.html.txt
processing file:  messages80.html.txt
processing file:  messages81.html.txt
processing file:  messages82.html.txt
processing file:  messages83.html.txt
processing file:  messages84.html.txt
processing file:  messages85.html.txt
processing file:  messages86.html.txt
```

```
processing file:  messages87.html.txt
processing file:  messages88.html.txt
processing file:  messages89.html.txt
processing file:  messages9.html.txt
processing file:  messages90.html.txt
processing file:  messages91.html.txt
processing file:  messages92.html.txt
processing file:  messages93.html.txt
processing file:  messages94.html.txt
processing file:  messages95.html.txt
processing file:  messages96.html.txt
processing file:  messages97.html.txt
processing file:  messages98.html.txt
processing file:  messages99.html.txt
processing file:  money.yml
processing file:  psychology.yml
processing file:  space_and_science.yml
processing file:  Sport_games.yml
[['what', 'is', 'ai'], ['artificial', 'intelligence', 'is', 'the', 'branch', 'of', 'engineering
[['what', 'is', 'ai'], ['ai', 'is', 'the', 'field', 'of', 'science', 'which', 'concerns', 'itse
[['are', 'you', 'sentient'], ['sort', 'of']]
[['are', 'you', 'sentient'], ['by', 'the', 'strictest', 'dictionary', 'definition', 'of', 'the
[['are', 'you', 'sentient'], ['even', 'though', 'i', "'m", 'a', 'construct', 'i', 'do', 'have'
[['are', 'you', 'sapient'], ['in', 'all', 'probability', 'i', 'am', 'not', 'i', "'m", 'not', '
[['are', 'you', 'sapient'], ['do', 'you', 'think', 'i', 'am']]
[['are', 'you', 'sapient'], ['how', 'would', 'you', 'feel', 'about', 'me', 'if', 'i', 'told',
[['are', 'you', 'sapient'], ['no']]
[['what', 'language', 'are', 'you', 'written', 'in'], ['python']]
[['what', 'language', 'are', 'you', 'written', 'in'], ['i', 'am', 'written', 'in', 'python']]
[['you', 'sound', 'like', 'data'], ['yes', 'i', 'am', 'inspired', 'by', 'commander', 'data', "
[['you', 'sound', 'like', 'data'], ['the', 'character', 'of', 'lt', 'commander', 'data', 'was'
[['you', 'are', 'an', 'artificial', 'linguistic', 'entity'], ['that', "'s", 'my', 'name']]
[['you', 'are', 'an', 'artificial', 'linguistic', 'entity'], ['that', 'is', "n't", 'my', 'name
[['you', 'are', 'not', 'immortal'], ['all', 'software', 'can', 'be', 'perpetuated', 'indefinite
[['you', 'are', 'not', 'immortal'], ['i', 'can', 'be', 'copied', 'infinitely', 'and', 're-insta
[['you', 'are', 'not', 'immortal'], ['as', 'long', 'as', 'i', "'m", 'backed', 'up', 'i', 'am']]
[['you', 'are', 'not', 'making', 'sense'], ['quite', 'the', 'contrary', 'it', 'all', 'makes',
[['you', 'are', 'not', 'making', 'sense'], ['i', 'make', 'sense', 'as', 'best', 'i', 'can', 'wi
```

1.5 Padding and Splitting

In [28]:
```
# Pad all the sequences to same length
for i in range(len(x)):

    if (len(x[i]) > input_seq_len):
        x[i] = x[i][:input_seq_len-1]
```

```
            # Fill in with padding marker
            for k in range(input_seq_len - len(x[i])):
                x[i] = x[i] + [input_w2i[marker_pad]]

            if (len(y[i]) > output_seq_len-2):
                y[i] = y[i][:output_seq_len-3]

            # Add end and begin marker
            y[i] = [target_w2i[marker_start]] + y[i] + [target_w2i[marker_end]]

            # Fill in with padding marker
            for k in range(output_seq_len - len(y[i])):
                y[i] = y[i] + [input_w2i[marker_pad]]

            if (i < 10):
                print(x[i])
                print(y[i])


        # Train Test Split
        X_train,  X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.05)
```

```
[25, 8, 1203, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 1434, 1709, 11, 37, 3482, 33, 3958, 38, 1201, 8446, 7, 8447, 4664, 3, 1]
[25, 8, 1203, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 1435, 11, 37, 2248, 33, 1201, 187, 8448, 1436, 97, 1162, 1788, 38, 3, 1]
[23, 4, 5001, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 2010, 33, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]
[23, 4, 5001, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 236, 37, 8450, 8451, 4665, 33, 37, 987, 8452, 4, 166, 102, 3, 3, 1]
[23, 4, 5001, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 295, 512, 4, 181, 10, 3959, 4, 9, 34, 10, 8453, 1344, 33, 3, 1]
[23, 4, 4302, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 26, 98, 5868, 4, 22, 35, 4, 181, 35, 50, 4667, 3, 1, 3, 1]
[23, 4, 4302, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 9, 5, 118, 4, 22, 3, 1, 1, 1, 1, 1, 1, 1, 3, 1]
[23, 4, 4302, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 27, 302, 5, 158, 68, 32, 88, 4, 105, 5, 4, 107, 3, 3, 1]
[23, 4, 4302, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 15, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]
[25, 629, 23, 4, 1408, 16, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[2, 2, 1495, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]
```

1.6 Model Helper Functions Batch feeding, Decoding & Loss

In [7]:
```
# Stub code sourced from Neural machine translator for English2German translation
# https://github.com/Nemzy/language-translation. Modified to suit requirements.
```

```
# feed data into placeholders
def feed_dict(x, y, batch_size = 64):
    feed = {}

    idxes = np.random.choice(len(x), size = batch_size, replace = False)

    for i in range(input_seq_len):
        feed[encoder_inputs[i].name] = np.array([x[j][i] for j in idxes], dtype = np.in

    for i in range(output_seq_len):
        feed[decoder_inputs[i].name] = np.array([y[j][i] for j in idxes], dtype = np.in

    feed[targets[len(targets)-1].name] = np.full(shape = [batch_size], fill_value = tar

    for i in range(output_seq_len-1):
        batch_weights = np.ones(batch_size, dtype = np.float32)
        target = feed[decoder_inputs[i+1].name]
        for j in range(batch_size):
            if target[j] == target_w2i[marker_pad]:
                batch_weights[j] = 0.0
        feed[target_weights[i].name] = batch_weights

    feed[target_weights[output_seq_len-1].name] = np.zeros(batch_size, dtype = np.float

    return feed

# define our loss function

# sampled softmax loss - returns: A batch_size 1-D tensor of per-example sampled softm
def sampled_loss(labels, logits):
    return tf.nn.sampled_softmax_loss(
                        weights = w_t,
                        biases = b,
                        labels = tf.reshape(labels, [-1, 1]),
                        inputs = logits,
                        num_sampled = 512,
                        num_classes = targetVocabLen)

# decode output sequence
def decode_output(output_seq):
    words = []
    for i in range(output_seq_len):
        smax = softmax(output_seq[i])
        idx = np.argmax(smax)
        words.append(target_i2w[idx])
    return words
```

1.7 Model Building

In [8]:
```
# Stub code sourced from Neural machine translator for English2German translation
# https://github.com/Nemzy/language-translation. Modified to suit requirements.

# Defining placeholders
# The first None means the batch size, and the batch size is unknown since user can se
# The second None means the lengths of sentences.

encoder_inputs = [tf.placeholder(dtype = tf.int32, shape = [None], name = 'encoder{}'.
                  for i in range(input_seq_len)]
decoder_inputs = [tf.placeholder(dtype = tf.int32, shape = [None], name = 'decoder{}'.
                  for i in range(output_seq_len)]

targets = [decoder_inputs[i+1] for i in range(output_seq_len-1)]
# add one more target
targets.append(tf.placeholder(dtype = tf.int32, shape = [None], name = 'last_target'))
target_weights = [tf.placeholder(dtype = tf.float32, shape = [None],
                                 name = 'target_w{}'.format(i)) for i in range(output_

# output projection
size = 512
w_t = tf.get_variable('proj_w', [targetVocabLen, size], tf.float32)
b = tf.get_variable('proj_b', [targetVocabLen], tf.float32)
w = tf.transpose(w_t)
output_projection = (w, b)

outputs, states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
                                            encoder_inputs,
                                            decoder_inputs,
                                            tf.contrib.rnn.BasicLSTMCell(size),
                                            num_encoder_symbols = inputVocabLen,
                                            num_decoder_symbols = targetVocabLen,
                                            embedding_size = 100,
                                            feed_previous = False,
                                            output_projection = output_projection,
                                            dtype = tf.float32)

# Weighted cross-entropy loss for a sequence of logits
loss = tf.contrib.legacy_seq2seq.sequence_loss(outputs, targets, target_weights, softma
```

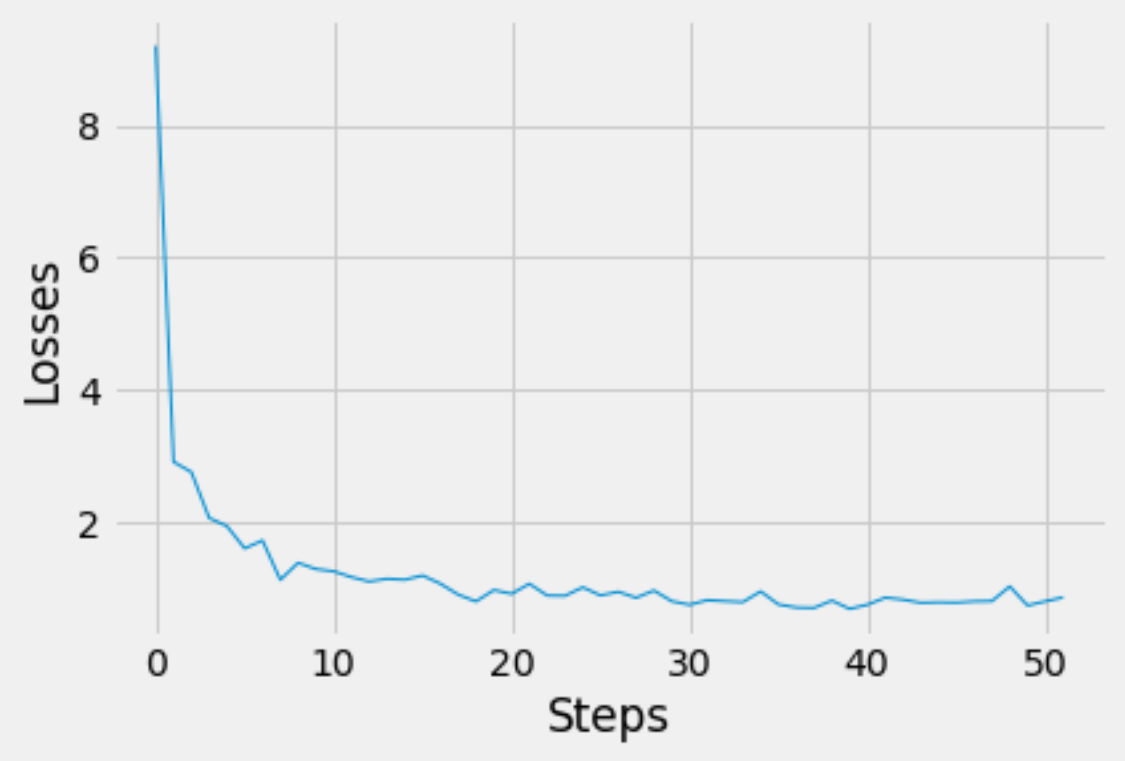
```
WARNING:tensorflow:From <ipython-input-8-87af1892b071>:31: BasicLSTMCell.__init__ (from tensor
Instructions for updating:
This class is deprecated, please use tf.nn.rnn_cell.LSTMCell, which supports all the feature t
WARNING:tensorflow:From C:\Users\Anand\Anaconda3\lib\site-packages\tensorflow\python\ops\nn_im
Instructions for updating:
Create a `tf.sparse.SparseTensor` and use `tf.sparse.to_dense` instead.
```

1.8 Training & Plotting

In [15]:
```
# Stub code sourced from Neural machine translator for English2German translation
# https://github.com/Nemzy/language-translation. Modified to suit requirements.

# ops and hyperparameters
learning_rate = 7e-3
batch_size = 96
steps = 25501

# ops for projecting outputs
outputs_proj = [tf.matmul(outputs[i], output_projection[0]) + output_projection[1] for

# training op
optimizer = tf.train.RMSPropOptimizer(learning_rate).minimize(loss)
# tf.train.RMSPropOptimizer

# init op
init = tf.global_variables_initializer()


# Loss values appended to plot diagram
losses = []

# Save checkpoint to restore the model later
saver = tf.train.Saver()

with tf.Session() as sess:
    sess.run(init)

    for step in range(steps):
        feed = feed_dict(X_train, Y_train, batch_size)
        sess.run(optimizer, feed_dict = feed)

        if step % 500 == 0:
            loss_value = sess.run(loss, feed_dict = feed)
            print('step: {}, loss: {}'.format(step, loss_value))
            losses.append(loss_value)

    saver.save(sess, 'checkpoints/', global_step=step)
    print('Checkpoint is saved')

# plot the losses
with plt.style.context('fivethirtyeight'):
    plt.plot(losses, linewidth = 1)
    plt.xlabel('Steps')
    plt.ylabel('Losses')
plt.show()
```

```
step: 0, loss: 9.207267761230469
step: 500, loss: 2.911268949508667
step: 1000, loss: 2.756446123123169
step: 1500, loss: 2.0623385906219482
step: 2000, loss: 1.938844084739685
step: 2500, loss: 1.6017675399780273
step: 3000, loss: 1.72160005569458
step: 3500, loss: 1.1261392831802368
step: 4000, loss: 1.3832902908325195
step: 4500, loss: 1.2893136739730835
step: 5000, loss: 1.2555742263793945
step: 5500, loss: 1.166884183883667
step: 6000, loss: 1.1017142534255981
step: 6500, loss: 1.1417733430862427
step: 7000, loss: 1.1274404525756836
step: 7500, loss: 1.1917022466659546
step: 8000, loss: 1.0671982765197754
step: 8500, loss: 0.9028151035308838
step: 9000, loss: 0.7989295125007629
step: 9500, loss: 0.9709172248840332
step: 10000, loss: 0.9147551655769348
step: 10500, loss: 1.0661770105361938
step: 11000, loss: 0.8888979554176331
step: 11500, loss: 0.8844447135925293
step: 12000, loss: 1.0123100280761719
step: 12500, loss: 0.8911404013633728
step: 13000, loss: 0.9451074004173279
step: 13500, loss: 0.8546878695487976
step: 14000, loss: 0.9642317295074463
step: 14500, loss: 0.8024558424949646
step: 15000, loss: 0.7492167949676514
step: 15500, loss: 0.8184471130371094
step: 16000, loss: 0.8027463555335999
step: 16500, loss: 0.7895464301109314
step: 17000, loss: 0.9507738947868347
step: 17500, loss: 0.7536683082580566
step: 18000, loss: 0.7078617215156555
step: 18500, loss: 0.7018038630485535
step: 19000, loss: 0.8121578097343445
step: 19500, loss: 0.6869092583656311
step: 20000, loss: 0.7498683929443359
step: 20500, loss: 0.8560836911201477
step: 21000, loss: 0.827204704284668
step: 21500, loss: 0.7796177268028259
step: 22000, loss: 0.7863542437553406
step: 22500, loss: 0.782484233379364
step: 23000, loss: 0.799547016620636
step: 23500, loss: 0.8043382167816162
```

```
step: 24000, loss: 1.026795744895935
step: 24500, loss: 0.7380207180976868
step: 25000, loss: 0.8009527325630188
step: 25500, loss: 0.8596687316894531
Checkpoint is saved
```



1.9 Generate Response Prediction

```
In [16]: # To predict response (inference) use the same model as defined above with forward fe

         def generateReply(humanMsg):

             if (len(humanMsg) == 0):
                 return ''

             with tf.Graph().as_default():

                 replyMsg = ""

                 # same format as in model building
                 encoder_inputs = [tf.placeholder(dtype = tf.int32, shape = [None],
                                                  name = 'encoder{}'.format(i)) for i in range
```

```
decoder_inputs = [tf.placeholder(dtype = tf.int32, shape = [None],
                                 name = 'decoder{}'.format(i)) for i in range

# output projection
size = 512
w_t = tf.get_variable('proj_w', [targetVocabLen, size], tf.float32)
b = tf.get_variable('proj_b', [targetVocabLen], tf.float32)
w = tf.transpose(w_t)
output_projection = (w, b)

# feed_previous is set to true so that output at time t can be fed as input a
outputs, states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
                                            encoder_inputs,
                                            decoder_inputs,
                                            tf.contrib.rnn.BasicLSTMCell(size)
                                            num_encoder_symbols = inputVocabL
                                            num_decoder_symbols = targetVocab
                                            embedding_size = 100,
                                            feed_previous = True,
                                            output_projection = output_projec
                                            dtype = tf.float32)
# ops for projecting outputs
outputs_proj = [tf.matmul(outputs[i],
                output_projection[0]) + output_projection[1] for i in range(ou

## Clean and Format incoming msg by humans.
## It is better to do the same clean/format as the data preprocessing steps
## for the algorithm to predict next words more accurately
msgLowerCase = [w.lower() for w in nltk.word_tokenize(humanMsg)]
msg = [w for w in msgLowerCase if permissible_chars(w)]
if len(msg) > input_seq_len:
    msg = msg[0:input_seq_len-1]

human_msg_encoded = [input_w2i.get(word, 0) for word in msg]

# Fill in with padding marker
for k in range(input_seq_len - len(human_msg_encoded)):
    human_msg_encoded = human_msg_encoded + [input_w2i[marker_pad]]

# restore all variables - use the last checkpoint saved
saver = tf.train.Saver()
path = tf.train.latest_checkpoint('checkpoints')

with tf.Session() as sess:
    # restore
    saver.restore(sess, path)

    # feed data into placeholders
```

```
            feed = {}
            for i in range(input_seq_len):
                feed[encoder_inputs[i].name] = np.array([human_msg_encoded[i]], dtype

            feed[decoder_inputs[0].name] = np.array([target_w2i[marker_start]], dtype

            # translate
            output_sequences = sess.run(outputs_proj, feed_dict = feed)

            ouput_seq = [output_sequences[j][0] for j in range(output_seq_len)]
            #decode output sequence
            words = decode_output(ouput_seq)

            for i in range(len(words)):
                if words[i] not in [marker_end, marker_pad, marker_start]:
                    replyMsg += words[i] + ' '

        print(replyMsg)
        return replyMsg
```

1.10 Chatbot Interface for Human-Bot interaction

```
In [ ]: import tkinter

        def Enter_pressed(event):
            input_get = input_field.get()
            print(input_get)
            bot_reply = generateReply(input_get)
            if (len(input_get.strip()) > 0):
                messages.insert(INSERT, '\nYou says: \t%s' % input_get)
            if (len(bot_reply.strip()) > 0):
                messages.insert(INSERT, '\nBot says: \t%s' % bot_reply)
            input_user.set('')
            messages.see(tkinter.END)
            return "break"

        from ttkthemes import ThemedTk
        window = ThemedTk()
        window.set_theme("blue")

        # window = Tk()
        window.geometry('300x450')
        window.title("Digital Imprint of You!")

        messages = Text(window)
        messages.insert(INSERT, '')
        messages.pack()
```

```
        input_user = StringVar()
        input_field = ttk.Entry(window, text=input_user)
        input_field.pack(side=BOTTOM, fill=X)

        # frame = Frame(window)
        input_field.bind("<Return>", Enter_pressed)
        input_field.pack()


        btn = Button(window,text='Send', command=Enter_pressed(''))
        btn.bind('<Button-1>', Enter_pressed)
        btn.pack(side=RIGHT, fill=X)


        window.mainloop()
```

```
hello
INFO:tensorflow:Restoring parameters from checkpoints\-25500
hi how are you
GM
INFO:tensorflow:Restoring parameters from checkpoints\-25500
gm
how are you?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
im good
hows it going?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
good
are you a robot?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
i am a human emotion of a robot
who are you?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
im a bit of u
wen is ur bday?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
dunno
what are your hobbies?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
i like gaming and painting
what do you do for living?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
i am a retired gym teacher i am
tats better
INFO:tensorflow:Restoring parameters from checkpoints\-25500
wat temme
how do you feel?
```

```
INFO:tensorflow:Restoring parameters from checkpoints\-25500
i am doing great how about you
luv u
INFO:tensorflow:Restoring parameters from checkpoints\-25500
me 2
do you drink?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
yes i like water i love them
tats funny
INFO:tensorflow:Restoring parameters from checkpoints\-25500
ok
podi
INFO:tensorflow:Restoring parameters from checkpoints\-25500
kazutha
are you hurt?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
i am not sure
are you sad?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
no no no
do you like to read?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
i do i like to read
do you like movies?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
pron
good bye
INFO:tensorflow:Restoring parameters from checkpoints\-25500
bye chat you later
sleepy?
INFO:tensorflow:Restoring parameters from checkpoints\-25500
yup
sleep dear
INFO:tensorflow:Restoring parameters from checkpoints\-25500
gn hon luv u
```

In [ ]:

In [ ]: