CharRNNMovieDialog

April 15, 2020

Personal mid-term small project

Character based RNN to generate movie dialog

Overview

I tried to use the public dataset from Cornell Movie-Dialogs Corpus to generate movie dialogs automatically.

Since I am very interested in the character based neural network, I tried to use char embedding and RNN(LSTM) to do the language model.

First we extract data and get vocabulary, since it is char based, the vocab only contains 82 chars.

The framework is very similar as we used in Homework 4, except we generate char by char.

Result

The model architechure is one embedding layer, followed by two LSTM layers with 128 units, then a dense layer with softmax activation.

I used 200k sentences to train, each sentences contains average 100 chars. See if we can generate meaningful sentences.

The perplexity decreased from 5 to about 2. So the training makes the model has some generative power. First, if we use "stateless" models, then it could not generate meaning sentences, and it actually can get the first captial letters correctly. Then if we use "stateful" models, then we could get meaning words even sentences, and a lof of the punctuations are correct. So the model actually learning the relationship between English chars very nicely!

To be improved

It is very promising if more time were given and more data is used to train. Also, after I generated meaningful words and sentences, I will try to embedding movie genre and movie cast position to generate more accurate dialogs.

In [1]:
```
# coding=utf-8
import keras
import json
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import csv, string
from collections import Counter
import os, sys, random, re
from sklearn.model_selection import train_test_split
```

```
Using TensorFlow backend.
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:516:
```

```
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:517:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:518:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:519:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:520:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/usr/local/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

1 The dataset

- 220,579 conversational exchanges between 10,292 pairs of movie characters
- involves 9,035 characters from 617 movies
- in total 304,713 utterances
- movie metadata included:
  - genres
  - release year
  - IMDB rating
  - number of IMDB votes
  - IMDB rating
- character metadata included:
  - gender (for 3,774 characters)
  - position on movie credits (3,321 characters)

B) Files description:

In all files the field separator is " +++$+++ "

- movie_titles_metadata.txt
  - contains information about each movie title
  - fields:
    - movieID,
    - movie title,
    - movie year,
    - IMDB rating,
    - no. IMDB votes,
    - genres in the format ['genre1','genre2',...,'genreN']
- movie_characters_metadata.txt
  - contains information about each movie character
  - fields:
    - characterID
    - character name
    - movieID
    - movie title
    - gender ("?" for unlabeled cases)
    - position in credits ("?" for unlabeled cases)
- movie_lines.txt
  - contains the actual text of each utterance
  - fields:
    - lineID
    - characterID (who uttered this phrase)
    - movieID
    - character name
    - text of the utterance
- movie_conversations.txt
  - the structure of the conversations
  - fields
    - characterID of the first character involved in the conversation

    * characterID of the second character involved in the conversation
    * movieID of the movie in which the conversation occurred
    * list of the utterances that make the conversation, in chronological order: ['lineID1','lineID2',...,'lineIDN'] has to be matched with movie_lines.txt to reconstruct the actual content
- raw_script_urls.txt
  - the urls from which the raw sources were retrieved

2 The purpose

This program is to generate movie dialog according to the movie genre, and character gender/position in the movie, maybe a "movie chatbot" is also an interesting application.

```
In [2]: if not os.path.exists('./movie-dialog-corpus/movie_lines_formatted.tsv'):
            with open('./movie-dialog-corpus/movie_lines.tsv', newline='\n') as csvfile1:
                with open('./movie-dialog-corpus/movie_lines_formatted.tsv','w') as csvfile:
                    writer = csv.writer(csvfile, delimiter='\t')
                    reader = csv.reader(csvfile1, delimiter='\t', quotechar=None)
                    for row in reader:
                        if row[0][0] == '"':
                            row[0] = row[0][1:]
                        if len(row) > 4:
                            row[4] = ' '.join(row[4:])
                            row = row[:5]
                        if row[4] == "":
                            continue
                        writer.writerow(row)
```

```
In [3]: movie_titles_metadata = pd.read_csv("./movie-dialog-corpus/movie_titles_metadata.tsv",
        encoding='utf-8-sig', sep="\t", header = None)
        movie_lines = pd.read_csv("./movie-dialog-corpus/movie_lines_formatted.tsv",
        encoding='utf-8-sig', sep="\t", header = None)
        movie_conversations = pd.read_csv("./movie-dialog-corpus/movie_conversations.tsv",
        encoding='utf-8-sig', sep="\t", header = None)
        movie_characters_metadata = pd.read_csv("./movie-dialog-
        corpus/movie_characters_metadata.tsv",  encoding='utf-8-sig', sep="\t", header = None)
```

```
In [4]: movie_lines.head()
```

[4]:

	0	1	2	3	4
0	L1045	u0	m0	BIANCA	They do not!
1	L1044	u2	m0	CAMERON	They do to!
2	L985	u0	m0	BIANCA	I hope so.
3	L984	u2	m0	CAMERON	She okay?
4	L925	u0	m0	BIANCA	Let's go.

```
In [5]: START = "<START>"
        UNK = "<UNK>"
        END = "<END>"
```

```
In [6]: def load_data_char():
            """
            Load the training data into the training format
            Return a list of characters
            """
            data = [list(h.strip('\n')) for h in movie_lines[movie_lines.columns[4]]]

            return data
```

```
        def load_data():
            """
            Load the training data into the training format
            remove punctuation and return a list of tokens
            """

            # Removing excess punctuation and newline
            pattern = re.compile('[%s]' % re.escape(string.punctuation))
            data = [pattern.sub('', h.strip("\n")).split(' ') for h in
        movie_lines[movie_lines.columns[4]]]

            return data
```

```
In [7]: vocab = []
        re_vocab = {}

        def get_vocab_char():
            lines = movie_lines[movie_lines.columns[4]]
            total_chars = ''.join(lines)
            total_chars = Counter(total_chars)
            vocab= [c for c in total_chars if total_chars[c] > 100 and (ord(c) < 120 or ord(c) >
        160) ]
            return vocab

        def get_vocab(min_token_ct=0):
            """
            For given training data, list of vocabulary list, i.g.
            [["this", "set", "1"],
             ["this", "is", "another", "set"],
             ]

            return the vocab list and rev_vocab dictionary
            3 numerical encodings are reserved: {<UNK>:0, <START>:1, <END>:2}
            """
            lines = load_data()
            token_ct = Counter([token for line in lines for token in line])
            token_ct = {k: v for k, v in token_ct.items() if v >= min_token_ct}
            vocab = sorted(token_ct, key=token_ct.get, reverse=True)

            return vocab
```

```
In [8]: vocab = get_vocab_char()
        vocab = [UNK, START, END] + vocab
        re_vocab = {v:k for k,v in enumerate(vocab)}
        print(len(re_vocab),re_vocab)
```

```
82 {'<UNK>': 0, '<START>': 1, '<END>': 2, 'T': 3, 'h': 4, 'e': 5, ' ': 6, 'd': 7, 'o':
8, 'n': 9, 't': 10, '!': 11, 'I': 12, 'p': 13, 's': 14, '.': 15, 'S': 16, 'k': 17,
'a': 18, '?': 19, 'L': 20, "'": 21, 'g': 22, 'W': 23, 'w': 24, 'O': 25, '-': 26, 'u':
27, 'r': 28, 'l': 29, 'i': 30, 'N': 31, 'm': 32, 'Y': 33, 'j': 34, 'b': 35, 'c': 36,
'"': 37, 'A': 38, 'q': 39, 'f': 40, 'v': 41, 'G': 42, 'M': 43, '9': 44, '0': 45, '2':
46, '1': 47, 'B': 48, 'H': 49, 'C': 50, 'P': 51, 'J': 52, 'E': 53, 'D': 54, 'F': 55,
'R': 56, 'K': 57, 'U': 58, 'Q': 59, ':': 60, 'V': 61, '*': 62, '7': 63, '5': 64, '$':
65, '3': 66, '6': 67, '4': 68, '8': 69, '/': 70, ';': 71, 'Z': 72, 'X': 73, ',': 74,
'<': 75, '>': 76, '&': 77, '[': 78, ']': 79, '_': 80, 'í': 81}
```

```
In [9]: from keras.utils import to_categorical
        def to_label(token):
            label = re_vocab.get(token, re_vocab[UNK])
            return label
```

```
In [10]: def generate_dialog(data, batch_size=100, one_hot = True):
             # one dialog contains movie genre(a vector of size_g), vector of characters
         gender/position, and their dialog, tokenized into single chars.
             # should be a generator to save memory
             # should be LSTM model to preserve long and short memory
```

```
            while True:
                # Shuffle the data so data order is different for different epochs
                random.shuffle(data)
                #X: np.array(batch_size, sent_len, embedding_dim)
                #Y: np.array(batch_size, sent_len, )
                X, y = [], []
                for s in data:
                    X.append([to_label(START)] + [to_label(t) for t in s])
                    y.append([to_label(t) for t in s] + [to_label(END)])
                    if len(X) >= batch_size:
                        X = pad_sequences(sequences=X, maxlen=sent_len, padding='post',
        value=to_label(END))
                        y = pad_sequences(sequences=y, maxlen=sent_len, padding='post',
        value=to_label(END))
                        if one_hot:
                            #X = to_categorical(X, num_classes=len(re_vocab))
                            y = to_categorical(y, num_classes=len(re_vocab))
                        yield X, y

                        X, y = [], []
```

In [11]:
```
def sample_with_weight(prob):
    unk_idx = re_vocab[UNK]
    prob[unk_idx] = 0 # Make sure we do not use UNK in the generated text
    if prob.sum() <= 0:
        prob[1:] = 1.0
    return np.random.choice(range(len(prob)), p=prob/prob.sum())
```

In [12]:
```
sent_len = 100 #max([len(s) for s in train_X]) + 1
size_c = 300
```

In [13]:
```
embeddings_dict = {}
#with open("glove.6B/glove.6B.100d.txt", 'r') as f:
with open("char-embeddings.txt", 'r') as f:
    for line in f:
        values = line.split()
        word = values[0]
        vector = np.asarray(values[1:], "float32")
        embeddings_dict[word] = vector
embedding_matrix = np.zeros((len(vocab), size_c))
for word, i in re_vocab.items():
    embedding_vector = embeddings_dict.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

In [14]:
```
from keras.layers import Dense, Embedding, LSTM, Activation, TimeDistributed, Reshape,
Bidirectional
from keras.models import Sequential
from keras.preprocessing.sequence import pad_sequences

# For simplicity, we use the embedding of words to feed the model, therefore
# no need to add a Embedding layer in the begining. But for a possibly better
performance
# you can add a embedding layer, even better if you use the glove embedding matrix as
the
# initial value for the embedding layer
# This is useful also because we have filled the embedding with random values for those
missing
# vocabularies, allowing the embedding matrix to relax during training will improve the
performance
# for these words as well. But be prepared that this would slow down the training

# Unfortunately Keras does not have an easy way to support dynamic length of input for
RNN model.
# So we use the sent_len to truncate all the sentences.
batch_size = 10
train_model = Sequential()
```

```
        train_model.add(Embedding(input_dim = len(vocab), output_dim = size_c, input_length =
        sent_len, trainable = True))
        train_model.add(LSTM(128, input_shape=(sent_len, len(vocab)), return_sequences=True))
        train_model.add(LSTM(128, input_shape=(sent_len, len(vocab)), return_sequences=True))
        train_model.add(TimeDistributed(Dense(len(vocab), activation='softmax')))
        train_model.summary()
```

```
WARNING: Logging before flag parsing goes to stderr.
W0412 19:00:51.767338 4716780992 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:74: The
name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph
instead.

W0412 19:00:51.794117 4716780992 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:517: The
name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

W0412 19:00:51.800951 4716780992 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:4138: The
name tf.random_uniform is deprecated. Please use tf.random.uniform instead.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 300)          24600
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 128)          219648
_________________________________________________________________
lstm_2 (LSTM)                (None, 100, 128)          131584
_________________________________________________________________
time_distributed_1 (TimeDist (None, 100, 82)           10578
=================================================================
Total params: 386,410
Trainable params: 386,410
Non-trainable params: 0
_________________________________________________________________
```

```
In [15]: pred_model = Sequential()
         pred_model.add(Embedding(input_dim = len(vocab), output_dim = size_c,
         batch_input_shape=(1, 1), trainable = True))
         pred_model.add(LSTM(128, input_shape=(1, len(vocab)), return_sequences=True, stateful =
         True))
         pred_model.add(LSTM(128, input_shape=(1, len(vocab)), return_sequences=True, stateful =
         True))
         pred_model.add(TimeDistributed(Dense(len(vocab), activation='softmax')))
         pred_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (1, 1, 300)               24600
_________________________________________________________________
lstm_3 (LSTM)                (1, 1, 128)               219648
_________________________________________________________________
lstm_4 (LSTM)                (1, 1, 128)               131584
_________________________________________________________________
time_distributed_2 (TimeDist (1, 1, 82)                10578
=================================================================
Total params: 386,410
```

```
Trainable params: 386,410
Non-trainable params: 0
_________________________________________________________________
```

```
In [16]: def generate_text(model, max_len=sent_len-1, seed=None):
             if seed is None:
                 seed = START
                 result = []
             else:
                 result = [seed]
             model.reset_states()
             for i in range(max_len):
                 X = [[to_label(seed)]]
                 #X = to_categorical(X, num_classes=len(re_vocab))
                 idx = sample_with_weight(model.predict(X)[0][0])
                 if vocab[idx] in [START, UNK, END]:
                     break
                 seed = vocab[idx]
                 result.append(seed)
             return ''.join(result)  # char based
             #return ' '.join(result) # word based
```

```
In [17]: generate_text(pred_model, seed='a')
```

```
W0412 19:00:53.262310 4716780992 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:174: The
name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_session
instead.

W0412 19:00:53.264347 4716780992 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:181: The
name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.
```

```
[17]: 'acZ>>EN]'
```

```
In [18]: def genre_embedding(size_g):
             pass
```

```
In [19]: def char_embedding(size_c):
             pass
```

```
In [20]: def calculate_perplexity(model, X, y):
             #prob = model.predict(to_categorical(X, len(vocab)))
             prob = model.predict(X)
             M = 0
             P = 0
             N, sent_len = y.shape
             for s in range(N):
                 for l in range(sent_len):
                     if y[s,l] in [re_vocab[END]]:
                         break
                     P += np.log(prob[s,l,y[s,l]])
                     M += 1
             perplexity = np.exp(-P/M)
             return perplexity
```

```
In [21]: train_X, dev_X = train_test_split(load_data_char(), test_size=0.33, shuffle = True)
         dev_X, dev_y = next(generate_dialog(dev_X, batch_size=-1, one_hot = False))
         train_X = train_X
```

```
In [22]: from keras.optimizers import Adam
         def on_epoch_end(epoch, logs):
             pred_model.set_weights(train_model.get_weights())
             print('----- Generating text after Epoch: %d' % (epoch + 1))
```

```
            for i in range(5):
                print(generate_text(pred_model))
            print('Current perplexity on dev data: ',
                  calculate_perplexity(train_model, dev_X, dev_y), '\n')

        from keras.callbacks import LambdaCallback
        """
        Notice how the metrics / generated text evolve after each epoch
        """
        batch_size = 10
        num_batches = len(train_X) // batch_size
        adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
        train_model.compile(loss='categorical_crossentropy', optimizer=adam)
        train_model.fit_generator(generate_dialog(train_X, batch_size), num_batches, 10,
                  callbacks=[LambdaCallback(on_epoch_end=on_epoch_end)]
                )
```

```
W0412 19:00:56.583560 4716780992 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/site-packages/keras/optimizers.py:790: The name
tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

W0412 19:00:56.786028 4716780992 deprecation.py:323] From
/usr/local/lib/python3.6/site-packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where


Epoch 1/10
20398/20398 [==============================] - 3547s 174ms/step - loss: 0.7109
----- Generating text after Epoch: 1
Oh I'm on impoiss close.
You didn't hope this perfect seak.
I forgot another airman.
Well because I had staiding an opant in for us. You dear…
Taking new of the job?
Current perplexity on dev data:  4.028639752313307

Epoch 2/10
20398/20398 [==============================] - 3446s 169ms/step - loss: 0.6171
----- Generating text after Epoch: 2
Your mother. We can unlosal foors?
Goodboe that another daumber in the maish.
Make a miles are comes.
Aren't that fuckin' loader?
It's noan in the free number make.
Current perplexity on dev data:  4.249186407965177

Epoch 3/10
20398/20398 [==============================] - 3192s 156ms/step - loss: 0.6028
----- Generating text after Epoch: 3
Neighment Gaboo.
Cool and I was going to then --
What does the familioon?'
How'd --
Well since I turrened to the important stupidaced.
Current perplexity on dev data:  3.5413985658892724

Epoch 4/10
20398/20398 [==============================] - 3278s 161ms/step - loss: 0.5955
```

```
----- Generating text after Epoch: 4
How girl. Calleus who would give me a week?
Hello Life these pressure.
This shakes in pardin killed Tockled.
Look I've told then. College.
You been' heart of Zaul?
Current perplexity on dev data:  3.6433960554576776

Epoch 5/10
20398/20398 [==============================] - 3143s 154ms/step - loss: 0.5907
----- Generating text after Epoch: 5
Yes.  Did not believe the law of work to CTWELWAK united Jimme.
If the other mission Javishs and falls to trafeling. You loved his mother?
You don't understand!
Listen.
Mernettia.  Spirit I have.
Current perplexity on dev data:  3.6165832922578116

Epoch 6/10
20398/20398 [==============================] - 3099s 152ms/step - loss: 0.5874
----- Generating text after Epoch: 6
What's not?
Well it haddeading is what the 'im admit what he's still not the brain?
Could know eatell."
And do but she's funnite engid that studiots.
How man is that?
Current perplexity on dev data:  3.467740390727447

Epoch 7/10
20398/20398 [==============================] - 3107s 152ms/step - loss: 0.5849
----- Generating text after Epoch: 7
Roba Cars.
Seat again.
Cartred's mother not.
If that's more than the truth.
We…
Current perplexity on dev data:  3.7304578710895524

Epoch 8/10
20398/20398 [==============================] - 3104s 152ms/step - loss: 0.5829
----- Generating text after Epoch: 8
It's FEC:Yaing.
It is… three he saved me taw-nine.
That answens familia!
You're torn.   I haven't done me!
Would that be too sweetheart' best second this?
Current perplexity on dev data:  3.3627057264822744

Epoch 9/10
20398/20398 [==============================] - 6536s 320ms/step - loss: 0.5813
----- Generating text after Epoch: 9
Give him the motive dent on a taking basic reed? I should be embarraged.
Nobodautes. Shapper his pain chritch if a sure on real enuails her coters.  You're
diffeing smashmo
What do we wib? What do I do?
I'm sure it starts and eat their stareds are leaved a great postsaust rings soon but
what happened?
There's coming to state and a world fremb and seems it.
Current perplexity on dev data:  3.3453610410392924
```

```
Epoch 10/10
20398/20398 [==============================] - 5088s 249ms/step - loss: 0.5799
----- Generating text after Epoch: 10
Oh much about a  the ware entering woman.
Can't have an operation.
You think we go coff he good sta make a cliful far?
What now?
Now later.
Current perplexity on dev data:  3.491889510312319
```

[22]: <keras.callbacks.History at 0x12fb15e48>

In [ ]: