assignment11_DavisAmie

February 18, 2021

1 File information

File: Assignment11.ipynb

Name: Amie Davis

Date: 2/17/2021

Course: DSC650 - Big Data

Assignment Number: 11

Purpose: Experiment with advanced deep learning use cases including text and image generation

2 References:

Chollet, F. (2018). Deep learning with Python. Shelter Island, NY: Manning Publications.

3 Assignment 11

Using section 8.1 in Deep Learning with Python as a guide, implement an LSTM text generator. Train the model on the Enron corpus or a text source of your choice. Save the model and generate 20 examples to the results directory.

[27]:
```
# Create corpus from files in enron folder
import os

file_dir = "enron"
corpus_file = "enron_corpos.txt"

with open(corpus_file, "w") as outfile:
    for root, dir, files in os.walk(file_dir):
        for file in files:
            with open(os.path.join(root, file), "r") as infile:
#                 print(infile)
                outfile.write(infile.read())
```

[32]:
```
# Convert corpus to lowercase
path = 'enron_corpos.txt'
orig_text = open(path).read().lower()
```

```
print('Original Corpus length:', len(orig_text))

# Truncate corpus to prevent memory allocation error
text = text[:1000000]
print('Truncated Corpus length:', len(text))
```

```
Original Corpus length: 34080525
Truncated Corpus length: 1000000
```

[53]:
```
# Vectorize sequence of characters

import numpy as np

maxlen = 60    # extract sequence of 60 characters
step = 3       # sample a new sequence every 3 characters

sentences = []
next_chars = []

for i in range(0, len(text) - maxlen, step):
    sentences.append(text[i: i + maxlen])
    next_chars.append(text[i + maxlen])

print('Number of sequences:', len(sentences))

# Lists unique characters in the corpus
chars = sorted(list(set(text)))
print('Unique characters:', len(chars))

# Maps unique characters to their index
char_indices = dict((char, chars.index(char)) for char in chars)

# One-hot Encode the characters
print('Vectorization...')
x = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        x[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1
```

```
Number of sequences: 333314
Unique characters: 66
Vectorization…
```

[54]:
```
# Create LSTM model to predict next character
```

```
import keras
from keras import layers

model = keras.models.Sequential()
model.add(layers.LSTM(128, input_shape=(maxlen, len(chars))))
model.add(layers.Dense(len(chars), activation='softmax'))
```

[55]:
```
# Compile model

optimizer = keras.optimizers.RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

[56]:
```
# Function to sample the next character given the model's predictions
def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

[57]:
```
# Generate Text
# Output results to file

import random
import sys
from contextlib import redirect_stdout

results_dir = "results"
output_file = os.path.join(results_dir, "output.txt")

# Change standard output to file
with open(output_file, 'w') as f:
    with redirect_stdout(f):

        # Trains the model for 20 epochs
        for epoch in range(1, 21):
            print('epoch', epoch)

            # Fits model for one iteration of data
            model.fit(x, y, batch_size=128, epochs=1)

            # Select a text seed at random
            start_index = random.randint(0, len(text) - maxlen - 1)
            generated_text = text[start_index: start_index + maxlen]
            print('--- Generating with seed: "' + generated_text + '"')
```

```
            # Tries a range of different sampling temperatures
            for temperature in [0.2, 0.5, 1.0, 1.2]:
                print('------ temperature:', temperature)
                sys.stdout.write(generated_text)

                # Generates 400 characters starting from the seed text
                for i in range(400):
                    sampled = np.zeros((1, maxlen, len(chars)))

                    # One hot-encodes the characters
                    for t, char in enumerate(generated_text):
                        sampled[0, t, char_indices[char]] = 1.

                    # Samples the next character
                    preds = model.predict(sampled, verbose=0)[0]
                    next_index = sample(preds, temperature)
                    next_char = chars[next_index]

                    generated_text += next_char
                    generated_text = generated_text[1:]

                    sys.stdout.write(next_char)
```

```
C:\Users\amomu\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
RuntimeWarning: divide by zero encountered in log
  after removing the cwd from sys.path.
```

[58]:
```
# Save model
model_file = os.path.join(results_dir, "enron_textgen.h5")

model.save(model_file)
```

[ ]:
```

```