Text_Gen

February 5, 2023

[1]:
```
# Read in a file with textual data
# build a model to generate new text from
# existing text.
# Christine J Orosco

# The dataset is the "Satirical News from the Onion" retrieved from the Kaggle␣
 ↪website.
# https://www.kaggle.com/datasets/undefinenull/satirical-news-from-the-onion
```

[1]:
```
import pandas as pd
import numpy as np
import keras
import re
```

[ ]:
```
## Read in the file
```

[2]:
```
#Load up dataset to extract just the Title and the Body
df1 = pd.read_csv('Onions News.csv', header = 0)
```

[ ]:
```
### Data Cleaning
```

[3]:
```
#Extract first 1000 rows from the subset

subset = df1.loc[0:500].copy()
```

[4]:
```
#Drop the unwanted column

subset.drop(['Published Time'], axis=1, inplace = True)
```

[5]:
```
# Convert entire dataframe to strings because the interpreter
# gave error msg expecting string and got float when I tried to concat title␣
 ↪and content columns

subset = subset.astype(str)
```

[6]:
```
#Concatenate the Title and Content columns and copy to dataframe
```

```
subset = pd.DataFrame(subset[['Title', 'Content']].apply(lambda x: " ".join(x),␣
 ↪axis = 1),\
                      columns = ['Content'])
```

[7]:
```
#Change strings to lower case

subset['Content'] = subset['Content'].str.lower()
```

[8]:
```
#remove punctuation

import string

def remove_punctuations(text):
    for x in string.punctuation:
        text = text.replace(x, '')
    return text

text = subset['Content'].apply(remove_punctuations)
```

[9]:
```
# write to csv file

text.to_csv('onions_text.csv')
```

[10]:
```
# Read flat file

with open('onions_text.csv') as f:
          text = f.read()
```

[ ]:
```
## Build the model
```

[ ]:
```
### Create sequences to input into Keras
```

[11]:
```
#Create sequences
# Length of extracted character sequences
maxlen = 60

# We sample a new sequence every `step` characters
step = 3

# This holds our extracted sequences
sentences = []

# This holds the targets (the follow-up characters)
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))
```

```
Number of sequences: 180026
```

[12]:
```
# List of unique characters in the corpus
chars = sorted(list(set(text)))
print('Unique characters:', len(chars))

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

```
Unique characters: 58
```

[ ]:
```
# Need to convert characters into binary arrays
```

[13]:
```
# Next, one-hot encode the characters into binary arrays.

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
```

```
Vectorization…
```

[14]:
```
#Build the model

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'))
```

[15]:
```
# Use the categorical features optimizer

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

[16]:
```
#Reweight original probability distribution and sample a character index

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)
```

[ ]:
```
## Training the Model
```

[ ]:
```
import os
import sys
import csv
```

[18]:
```
# Define results directory for model output

results_dir = '~/assignments'

results_file = os.path.join(results_dir,'results_file.txt')
```

[ ]:
```
# Train the model and then generate new text
# I ran the model on Google Colab and read the text from the newly created text␣
 ↪file onions_text.csv.
# I did not go through loading the original text, create the Df, and select the␣
 ↪columns since I had run this on the
# VM at the University. The VM at Bellevue eventually quit.  So I went to␣
 ↪Google Colab.
# After 14 epochs, the model quit.  I think my VM ran out of memory. 38 minutes␣
 ↪lapsed before the model quit.
# I saved the results to the results file

import random
import sys

for epoch in range(1, 20):
    print('epoch', epoch)
    # Fit the model for 1 epoch on the available training 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 + '"')

    for temperature in [0.2, 0.5, 1.0, 1.2]:
        print('------ temperature:', temperature)
        sys.stdout.write(generated_text)

        # We generate 200 characters
```

```
        for i in range(200):
            sampled = np.zeros((1, maxlen, len(chars)))
            for t, char in enumerate(generated_text):
                sampled[0, t, char_indices[char]] = 1.

            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)
            sys.stdout.flush()

        print()
```

[ ]: