Assessment Work Report

TEXT GENERATION USING NLP

Project submitted in fulfilment of the requirements for the internship

Data Science (NLP) Internship

in

KUDOSWARE

Submitted By

Narsipalli Nagasatya Srinivasa Nikhilendra

nikhilnarsipalli@gmail.com

https://www.linkedin.com/in/nikhil-narsipalli-16904b233/

+91 7893051246

DECLARATION STATEMENT

I hereby declare that the assessment work reported in the dissertation/dissertation proposal entitled "TEXT GENERATION USING NLP" in partial fulfilment of the requirement for the Data Science(NLP) Internship in KUDOSWARE is an authentic. I have not submitted this work elsewhere for any degree or diploma.

Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Narsipalli Nagasatya Srinivasa Nikhilendra

TABLE OF CONTENTS

Text Generation Using NLP

Objective:

The main objective is building a model on Text Generation using NLP, on the subset we want to create an AI generated Tweet from the vast datasets of the Tweets that can be obtained.

Aim:

To build a Natural Language Processing model for the objective of tweets generation.

Dataset:

For this assessment, the dataset "Sentiment140 dataset" is being used which contains 1.6 million tweets extracted using twitter API. The dataset has 6 fields namely target, ids, date, flag, user, text. Out of these fields, we will be using text field – which contains the 1.6 million tweets.

Libraries Used:

- NumPy:
    - NumPy is also known as Numerical Python
    - NumPy is used for numerical computing, particularly for working with arrays and matrices. It provides a set of powerful tools for data manipulation, mathematical operations, and statistical analysis. The main object in NumPy is the ndarray (n-dimensional array), which is a homogeneous array of elements of a single data type, such as integers or floating-point numbers.
- Pandas:
    - pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series.

- Tensorflow:

    - TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

- Keras Framework:

    - Keras is a high-level open-source deep learning framework that runs on top of other lower-level frameworks such as TensorFlow, Theano, and CNTK. It provides a user-friendly API that makes it easier to build and train deep neural networks.

    - Keras is widely used in industry and academia for a variety of deep learning applications, and is particularly popular in the fields of computer vision and natural language processing.

- NLTK:
    - This is also known as Natural language ToolKit.

    - NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. It supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

- RE:
    - This is also known as Regular Expression.

    - A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

Methodology:

- Importing Libraries:
  - import numpy as np
  - import pandas as pd
  - import tensorflow as tf
  - from keras.models import Sequential
  - from keras.layers import Dense
  - from keras.layeres import LSTM
  - from keras.callbacks import ModelCheckpoint
  - from keras.utils import to_categorical
  - from nltk.translate.bleu_score import SmoothingFunction, corpus_bleu, sentence_bleu
  - from nltk.corpus import stopwords
  - from nltk.tokenize import word_tokenize
  - from nltk.stem import WordNetLemmatizer
  - import re
  - from keras.utils import pad_sequences
  - from keras.layers import Dropout

Importing required libraries

```
[1]  import numpy as np
     import pandas as pd
     import tensorflow as tf
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.layers import LSTM
     from keras.callbacks import ModelCheckpoint
     from keras.utils import to_categorical

     from nltk.translate.bleu_score import SmoothingFunction, corpus_bleu, sentence_bleu
```

```
[105] from nltk.corpus import stopwords
      from nltk.tokenize import word_tokenize
      from nltk.stem import WordNetLemmatizer
      import re
      from keras.utils import pad_sequences
      from keras.layers import Dropout
```

- Downloading the missing libraries:

```
[4] import nltk
    nltk.download('punkt')

    import nltk
    nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
[5] import nltk
    nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

- Importing the dataset:

  - The dataset “Sentiment140” is downloaded from Kaggle and mounted on to Google drive. Now, importing that dataset from Google drive to Colab platform.

Imprting the data set

```
[7] main_file = pd.read_csv("/content/drive/MyDrive/colab files/training.1600000.processed.noemoticon.csv",encoding = 'latin-1')
```

- Checking the imported dataset:

  - Checking the head and tails of the dataset to understand how the data is maintained in a tabular form.

```
main_file.head()
```

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

```
[9] main_file.tail()
```

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
1599994	4	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best fee...
1599995	4	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interv...
1599996	4	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me f...
1599997	4	2193602064	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	tinydiamondz	Happy 38th Birthday to my boo of alll time!!! ...
1599998	4	2193602129	Tue Jun 16 08:40:50 PDT 2009	NO_QUERY	RyanTrevMorris	happy #charitytuesday @theNSPCC @SparksCharity...

- Now, checking the info of the data

```
main_file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599999 entries, 0 to 1599998
Data columns (total 6 columns):
 #   Column                                                                                                  Non-Null Count    Dtype
---  ------                                                                                                  --------------    -----
 0   0                                                                                                       1599999 non-null  int64
 1   1467810369                                                                                              1599999 non-null  int64
 2   Mon Apr 06 22:19:45 PDT 2009                                                                            1599999 non-null  object
 3   NO_QUERY                                                                                                1599999 non-null  object
 4   _TheSpecialOne_                                                                                         1599999 non-null  object
 5   @switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You shoulda got David Carr of Third Day to do it. ;D  1599999 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

- Let’s confirm the column name of the tweets column to further use it.

```
[12] tweet_column = "@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You shoulda got David Carr of Third Day to do it. ;D"
     main_file[tweet_column][0]
```

```
'is upset that he can't update his Facebook by texting it... and might cry as a result  School today also. Blah!'
```

- Shape of the data set

```
[13] main_file[tweet_column].shape
```

```
(1599999,)
```

- Checking few other attributes to understand the data

Finding the average number of characters in each tweet

```
[14] main_file[tweet_column].apply(len).mean()
```

```
74.09008568130355
```

```
[15] tweets = main_file[tweet_column]
```

checking for null values in the dataset

```
[16] pd.isnull(tweets).sum() #i.e no null values
```

```
0
```

- Defining a function to – Remove punctuations:

  - Here, we are using regular expressions to remove all punctuations from the input data except '@' character, as this character has significance in tweets.

defininf removing punctuations

```
[56] def remove_punctuation(sentence):
       no_punctuation = re.sub(r'[^\w\s]'+'@'+'',"", str(sentence))
       return no_punctuation
```

```
[57] remove_punctuation(tweets[1])
```

```
'@Kenichan I dived many times for the ball. Managed to save 50%  The rest go out of bounds'
```

- Defining a function to – Lemmatize tokens:

  - In this tokens variable is taken as input and lemmatized tokens are returned

Defining Lemmatizing tokens

```
[20] def tokens_lemmatizer(tokens):
       lemmatizer = WordNetLemmatizer()
       lemmatized_tokens = [lemmatizer.lemmatize(w) for w in tokens]
       return lemmatized_tokens
```

- Cleaning the tweets using lemmatizing and punctuation:

    - Here, we are calling the pre-defined functions to lemmatize the tokens and remove punctuations .

Cleaning tweets

```
[51] clean_tweets = []

     for x in tweets:

       clean_tweets_process = []

       clean_tweets_process = tokens_lemmatizer(remove_punctuation(x))

       clean_tweets.append(clean_tweets_process)
```

- Mapping the characters to integer values:

    - Here, we are indexing each character to an integer to later normalize them.

Maping characters of the corpus to numbers

```
text = ""
corpus = []

for x in clean_tweets:
  for y in x:
    text = text + str(y).lower()
    corpus.append(y)

chars = sorted(list(set(text.lower())))

char_to_int = dict((c, i) for i, c in enumerate(chars))
```

- Input and Output Characters:

    - Now, we are preparing input and output characters by using sequence length as 144 as the max tweet length is the same.

```
# preparing for input to output pairs encoded as integers
sequence_length = 144
dataX = []
dataY = []
for i in range(0, n_chars - sequence_length, 1):
 seq_in = text[i:i + sequence_length]
 seq_out = text[i + sequence_length]
 dataX.append([char_to_int[char] for char in seq_in])
 dataY.append(char_to_int[seq_out])
n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)
```

```
Total Patterns:  373515
```

- Normalizing, Reshaping Input, Hot Encoding:

```
[107] # reshape X to be [samples, time steps, features]
      X = np.reshape(dataX, (n_patterns, sequence_length, 1))

      # normalizing values to 0-1
      X = X / float(n_vocab)

      #hot encoding
      y = to_categorical(dataY)
```

- Building the model and Compiling:

    - Here, I am building a LSTM model of 5 layers consisting of 246 nodes and 2 LSTM layers, 2 Dropout layers and a Dense layer. Compiled with Adam optimizer.

```
[117] # define the LSTM model
      model = Sequential()
      model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2]), return_sequences=True))
      model.add(Dropout(0.2))
      model.add(LSTM(256))
      model.add(Dropout(0.2))
      model.add(Dense(y.shape[1], activation='softmax'))
      model.compile(loss='categorical_crossentropy', optimizer='adam')
```

- Creating a checkpoint:

    - A checkpoint is created to store the best weights of the model as a snapshot to recover them at any point of time.

```
[114] # creating checkpoints
      filepath="/content/drive/MyDrive/colab files/Tweet generator project/weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
      checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save_best_only=True, mode='min')
      callbacks_list = [checkpoint]
```

- Fitting the model:

    - The model was fitted for 3 epochs with batch size set to 128.

    - Lower number of epochs were set because of time constrains, but setting larger number of epochs and lower batch size helps model in being more accurate in text generation.

```
[120] # fitting the model
      model.fit(X, y, epochs=1, batch_size=128, callbacks=callbacks_list)

      2919/2919 [==============================] - ETA: 0s - loss: 2.4246
      Epoch 1: loss improved from 2.70897 to 2.42464, saving model to /content/drive/MyDrive/colab files/Tweet generator project/weights-improvement-01-2.4246.hdf5
      2919/2919 [==============================] - 7898s 3s/step - loss: 2.4246
      <keras.callbacks.History at 0x7f46f13d3ee0>

      model.fit(X,y, epochs = 1, batch_size = 128, callbacks = callbacks_list)

      2823/2919 [============================>.] - ETA: 4:09 - loss: 2.2708
```

- Reverse Mapping Integers to Characters:

    - For us to understand the output generated text we need to convert it back from integers to characters.

```
[122] # reverse mapping int to characters

      int_to_char = dict((i, c) for i, c in enumerate(chars))
```

- Defining a function – to generate text:

```
def create_tweets(length):

  import sys
  output = ""
  start = np.random.randint(0, len(dataX)-1)
  pattern = dataX[start]
  for x in pattern:
    output = output + str(int_to_char[x])

  # generate characters
  for i in range(length):
    x = np.reshape(pattern, (1, len(pattern), 1))
    x = x / float(n_vocab)
    prediction = model.predict(x, verbose=0)
    index = np.argmax(prediction)
    result = int_to_char[index]
    seq_in = [int_to_char[value] for value in pattern]
    output = output + str(result)
    pattern.append(index)
    pattern = pattern[1:len(pattern)]

  return output
```

- Testing the model:

```
[198] created = create_tweets(100)

[199] print(created)

paint aww rip your wallllllll justanotherjerk i wanna c quotno doubtquot soooo to sore to sork to see to sore to sork to see to sore to sork to see to sore to sork to see to sore
```

  - **Seed** → l cant believe you have to move i will be over after work to help paint
  - **Ouput** → l cant believe you have to move i will be over after work to help paint aww rip your wallllllll justanotherjerk i wanna c quotno doubtquot soooo to sore to sork to see to sore to sork to see to sore to sork to see to sore to sork to see to sore

Improvements:

- Due, to the time constraint the number of epochs for the model training were negligible. Increasing the number of epochs to 50 – 100 would vastly improve the accuracy of the output generated text.
- Increasing the number of nodes to 700 might also help.
- Decrease batch size while fitting the model.
- Adding more number of layers.

Conclusions:

- **The output produced is not grammatically correct nor is meaningful. This is because of the low number of cycles of training.**
- **The model can be made more accurate with the mentioned improvements.**

Link for the Code:

https://colab.research.google.com/drive/1i-QevYBBXkJKC5W3CJRsB85DzT-Vkdbc?usp=sharing