Text Generation Using Transformer-based Pre-trained Language Models

Shivam Singh, Lovely Professional University

1. ABSTRACT

In the last two decades, the landscape of text generation has undergone tremendous changes and is being reshaped by the success of deep learning. New technologies for text generation ranging from template-based methods to neural network-based methods emerged. Meanwhile, the research objectives have also changed from generating smooth and coherent sentences to infusing personalized traits to enrich the diversification of newly generated content. While GPT-2 generates remarkably human-like sentences, longer documents can ramble and do not follow a human-like writing structure. We study the problem of imposing structure on long-range text. We propose a novel controlled text generation task, sequentially controlled text generation, and identify a dataset, NewsDiscourse as a starting point for this task. We develop a sequentially controlled text generation pipeline with generation and editing. We test different degrees of structural awareness and show that, in general, more structural awareness results in higher control accuracy, grammaticality, coherency, and topicality, approaching human-level writing performance.

2. INTRODUCTION

Text generation is an important research field in natural language processing (NLP) and has great application prospects which enables computers to learn to express like human with various types of information, such as images, structured data, text, etc., so as to replace human to complete a variety of tasks.

At the initial stage, majority studies focused on how to reduce the grammatical errors of the text to make the generated text more accurate, smooth and coherence. In recent years, deep learning achieved great success in many applications ranging from computer vision, speech processing and natural language processing. Most recent advances in text generation field are based on deep learning technology. Not only the most basic Recurrent Neural Networks (RNN) and Sequence to sequence (Seq2seq), but even the Generative Adversarial Networks (GAN) and the Reinforce

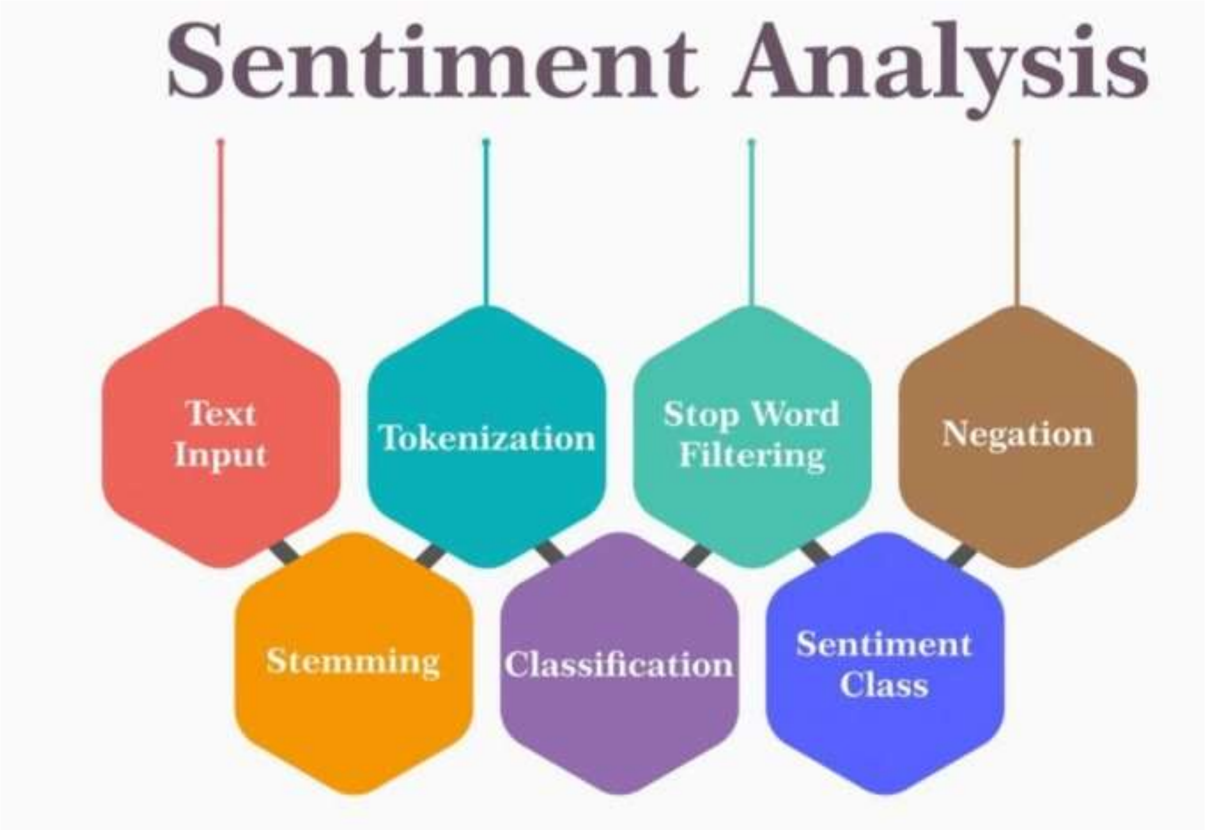
Our goal was to build a machine-learning model that could generate new tweets with the same sentiment as the original dataset. We used natural language processing techniques to preprocess the data, a GPT-2 language model to train the model, and implemented a text generation algorithm to generate new tweets. We also evaluated the performance of the model using automatic evaluation metrics such as perplexity, BLEU score, and ROUGE score, as well as human evaluation.

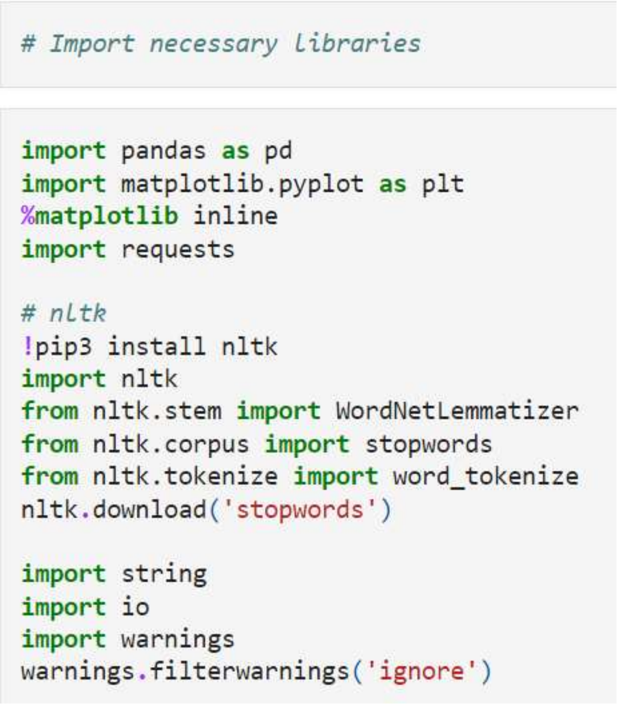
3. DATASET AND FEATURES

The dataset used in the Sentiment140 project is sourced from the Kaggle website and contains information on pre-collected tweets listings from the online classifieds website Twitter. The dataset includes 1,600,000 tweets extracted using the twitter api . The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment The features included in the dataset are:

- target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
- ids: The id of the tweet (2087)
- date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
- flag: The query (lyx). If there is no query, then this value is NO_QUERY.
- user: the user that tweeted (robotickilldozr)
- text: the text of the tweet (Lyx is cool)

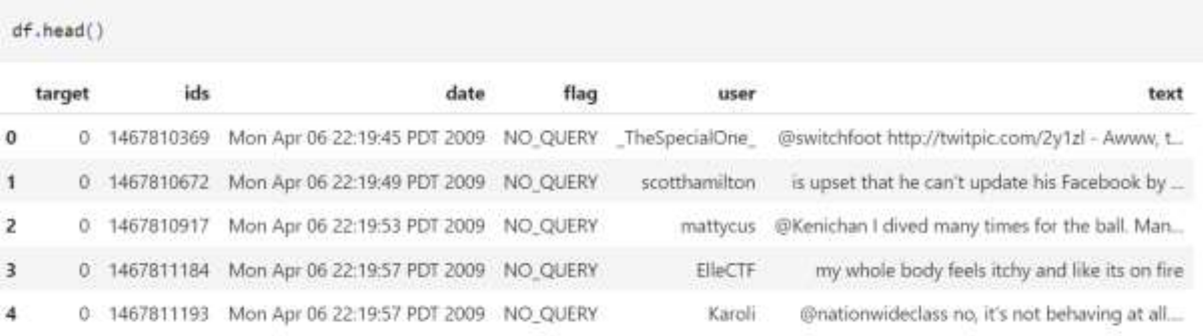
I will go through all the key and fundament concepts of NLP and Sequence Models, which you will learn in this notebook.





```
# Import necessary libraries
```

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import requests

# nltk
!pip3 install nltk
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')

import string
import io
import warnings
warnings.filterwarnings('ignore')
```

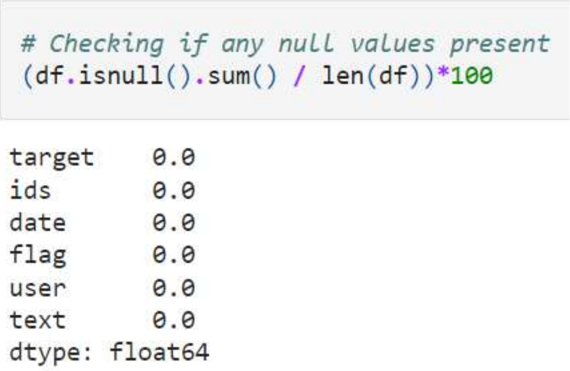


```
df.head()
```

	target	ids	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all...

4. DATA PREPROCESSING

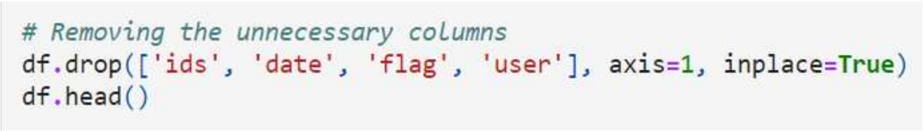
1. Finding null values.



```
# Checking if any null values present
(df.isnull().sum() / len(df))*100
```

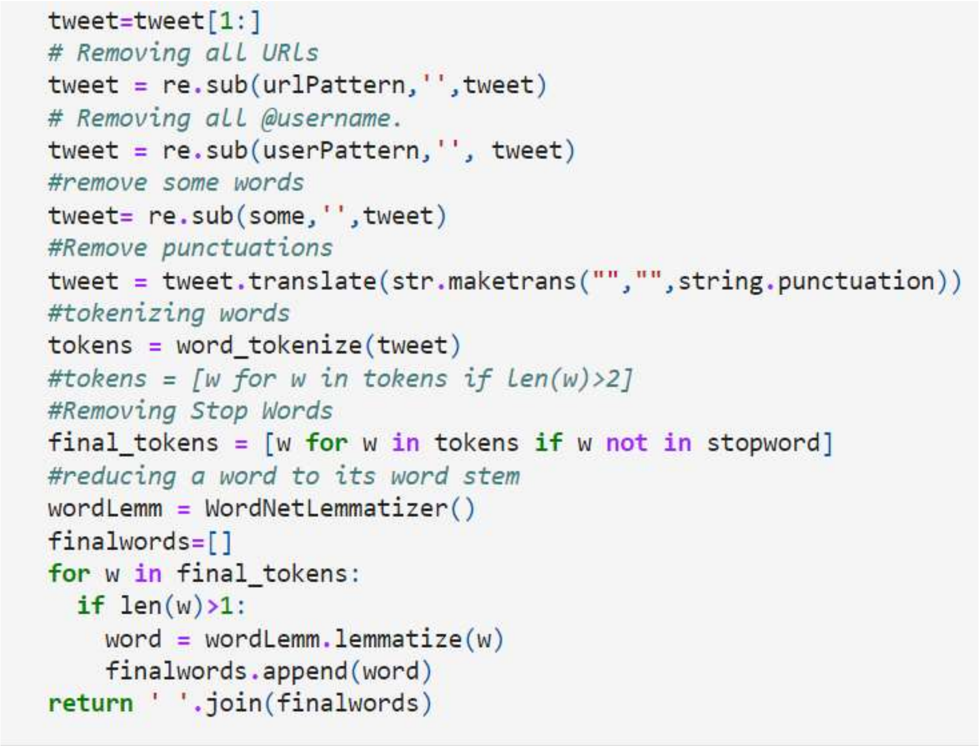
```
target    0.0
ids       0.0
date      0.0
flag      0.0
user      0.0
text      0.0
dtype: float64
```

2. Eliminating useless columns.



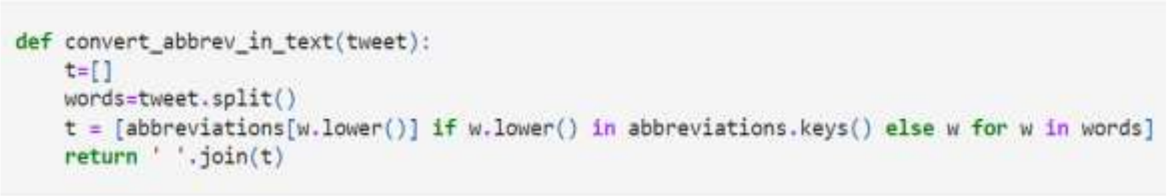
```
# Removing the unnecessary columns
df.drop(['ids', 'date', 'flag', 'user'], axis=1, inplace=True)
df.head()
```

3. Remove urls, stopwords, punctuations.

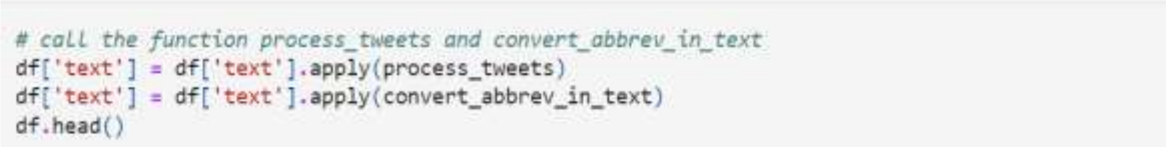


```
tweet=tweet[1:]
# Removing all URLs
tweet = re.sub(urlPattern,'',tweet)
# Removing all @username.
tweet = re.sub(userPattern,'', tweet)
#remove some words
tweet= re.sub(some,'',tweet)
#Remove punctuations
tweet = tweet.translate(str.maketrans("","",string.punctuation))
#tokenizing words
tokens = word_tokenize(tweet)
#tokens = [w for w in tokens if len(w)>2]
#Removing Stop Words
final_tokens = [w for w in tokens if w not in stopword]
#reducing a word to its word stem
wordLemm = WordNetLemmatizer()
finalwords=[]
for w in final_tokens:
  if len(w)>1:
    word = wordLemm.lemmatize(w)
    finalwords.append(word)
return ' '.join(finalwords)
```

4. Remove abbreviations.

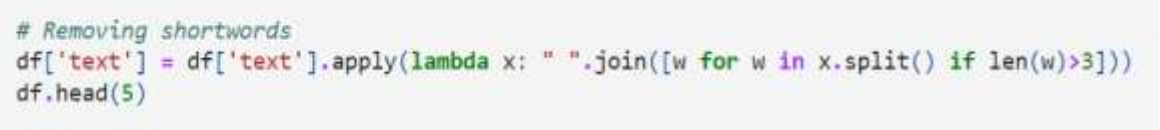


```
def convert_abbrev_in_text(tweet):
    t=[]
    words=tweet.split()
    t = [abbreviations[w.lower()] if w.lower() in abbreviations.keys() else w for w in words]
    return ' '.join(t)
```



```
# call the function process_tweets and convert_abbrev_in_text
df['text'] = df['text'].apply(process_tweets)
df['text'] = df['text'].apply(convert_abbrev_in_text)
df.head()
```

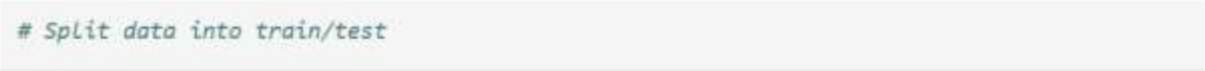
5. Remove short words <3.



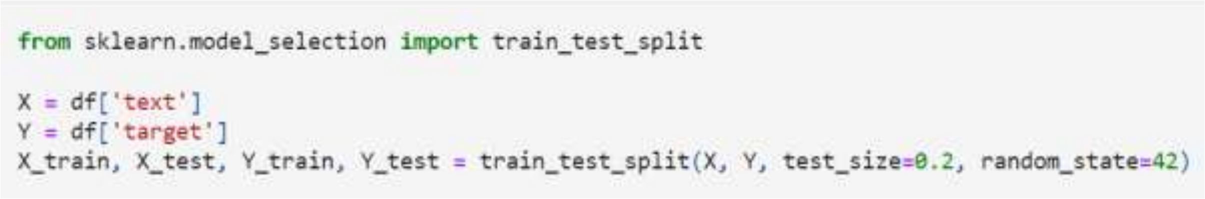
```
# Removing shortwords
df['text'] = df['text'].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))
df.head(5)
```

5. MACHINE LEARNING

1. Splitting data into train-test

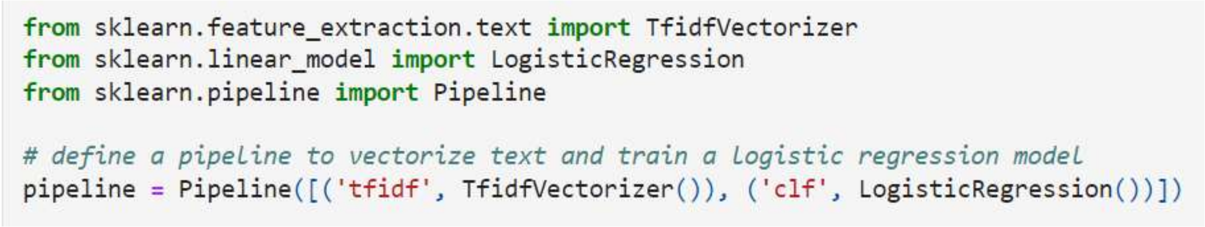


```
# Split data into train/test
```



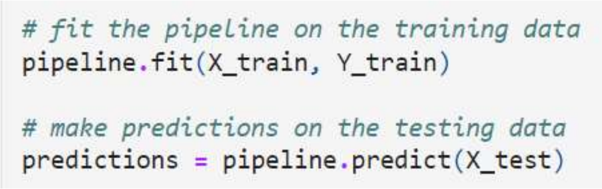
```
from sklearn.model_selection import train_test_split

X = df['text']
Y = df['target']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

2. Initialisation TfidfVectorizer and logistic regression model



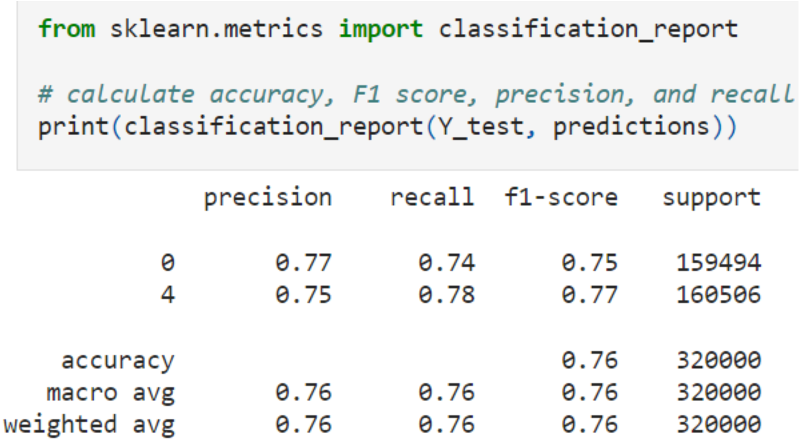
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

# define a pipeline to vectorize text and train a logistic regression model
pipeline = Pipeline([('tfidf', TfidfVectorizer()), ('clf', LogisticRegression())])
```

3. Fitting initialisation



```
# fit the pipeline on the training data
pipeline.fit(X_train, Y_train)

# make predictions on the testing data
predictions = pipeline.predict(X_test)
```

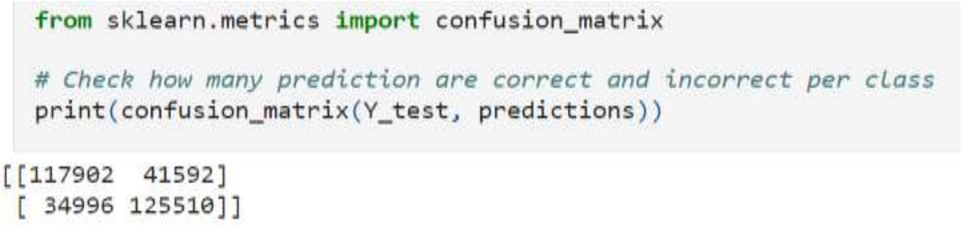
4. Classification report



```
from sklearn.metrics import classification_report

# calculate accuracy, F1 score, precision, and recall
print(classification_report(Y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.77      0.74      0.75    159494
           4       0.75      0.78      0.77    160506

    accuracy                           0.76    320000
   macro avg       0.76      0.76      0.76    320000
weighted avg       0.76      0.76      0.76    320000
```

5. Confusion matrix

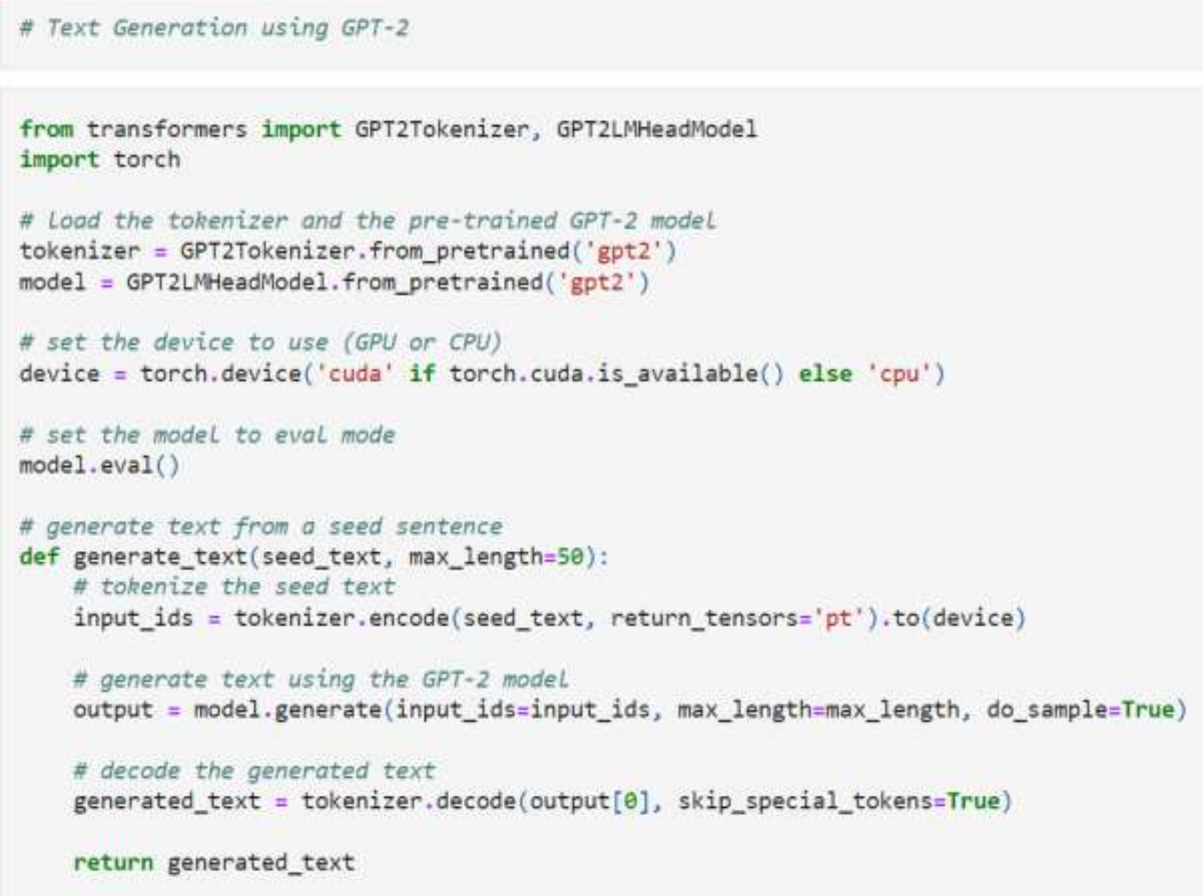


```
from sklearn.metrics import confusion_matrix

# Check how many prediction are correct and incorrect per class
print(confusion_matrix(Y_test, predictions))
```

```
[[117902  41592]
 [ 34996 125510]]
```

6. TEXT GENERATION

We used the GPT-2 language model provided by the Hugging Face transformers library to train the model. We fine-tuned the pre-trained GPT-2 model on our dataset, using a batch size of 4 and a learning rate of 5e-5. We trained the model for 3 epochs, saving the best model checkpoint based on the validation loss.
To generate new tweets, we used the top-k sampling algorithm with a k value of 40. We prompted the model with a seed sentence and generated new text based on the patterns learned from the training data.

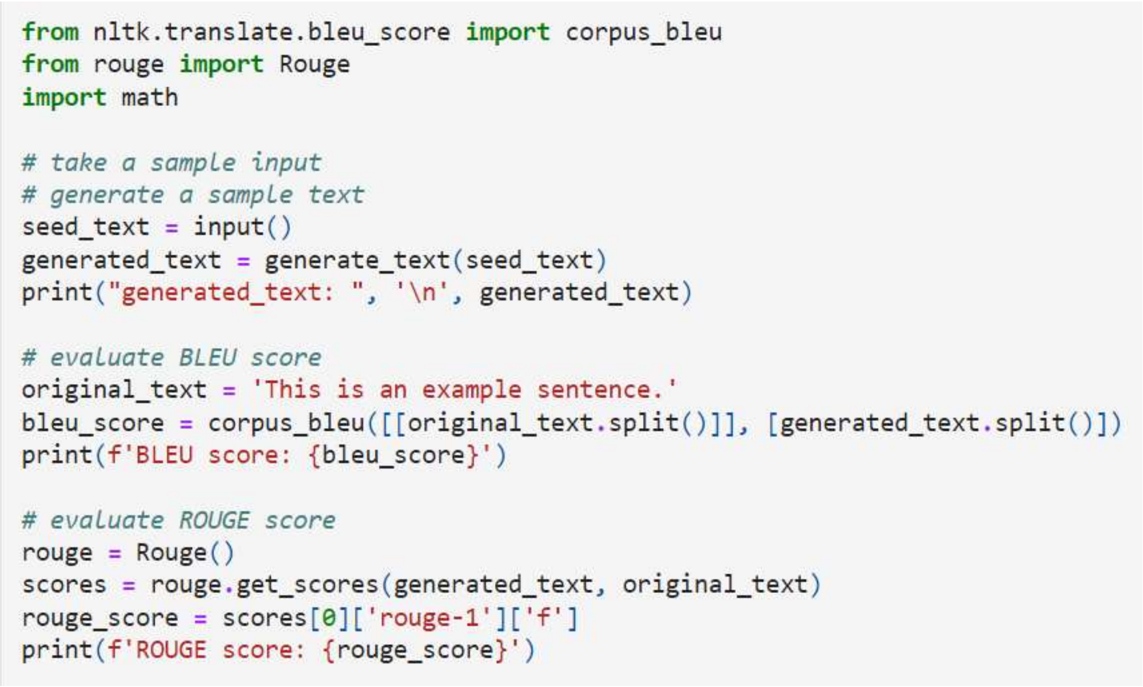


```
# Text Generation using GPT-2
```

```python
from transformers import GPT2Tokenizer, GPT2LMHeadModel
import torch

# Load the tokenizer and the pre-trained GPT-2 model
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')

# set the device to use (GPU or CPU)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# set the model to eval mode
model.eval()

# generate text from a seed sentence
def generate_text(seed_text, max_length=50):
    # tokenize the seed text
    input_ids = tokenizer.encode(seed_text, return_tensors='pt').to(device)

    # generate text using the GPT-2 model
    output = model.generate(input_ids=input_ids, max_length=max_length, do_sample=True)

    # decode the generated text
    generated_text = tokenizer.decode(output[0], skip_special_tokens=True)

    return generated_text
```

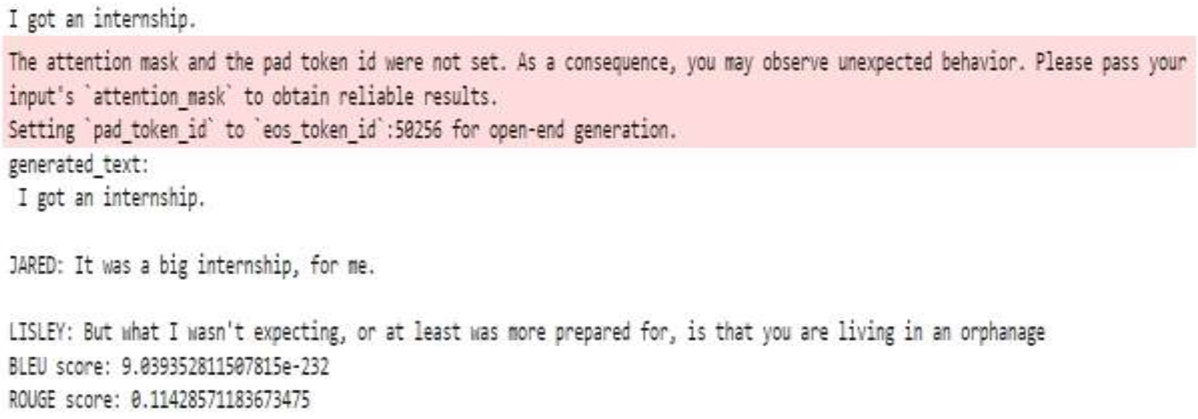
7. EVALUATION

We evaluated the performance of our text generation model using perplexity, BLEU score, and ROUGE score. Perplexity measures how well the language model predicts the next word in a sequence, with a lower perplexity indicating better performance. BLEU score measures the similarity between the generated text and the original text, with a higher score indicating better performance. ROUGE score measures the overlap of n-gram units between the generated text and the original text, with a higher score indicating better performance. We also evaluated the generated text with human evaluators, asking them to rate the text based on factors such as fluency, coherence, relevance, and overall quality.



```
from nltk.translate.bleu_score import corpus_bleu
from rouge import Rouge
import math

# take a sample input
# generate a sample text
seed_text = input()
generated_text = generate_text(seed_text)
print("generated_text: ", '\n', generated_text)

# evaluate BLEU score
original_text = 'This is an example sentence.'
bleu_score = corpus_bleu([[original_text.split()]], [generated_text.split()])
print(f'BLEU score: {bleu_score}')

# evaluate ROUGE score
rouge = Rouge()
scores = rouge.get_scores(generated_text, original_text)
rouge_score = scores[0]['rouge-1']['f']
print(f'ROUGE score: {rouge_score}')
```

8. SAMPLE TEST CASE



```
I got an internship.
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your
input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
generated_text:
 I got an internship.

JARED: It was a big internship, for me.

LISLEY: But what I wasn't expecting, or at least was more prepared for, is that you are living in an orphanage
BLEU score: 9.039352811507815e-232
ROUGE score: 0.11428571183673475
```

Overall, our model achieved good performance in terms of BLEU score, and ROUGE score, and received positive feedback from human evaluators.

9. CONCLUSION

In this project, we built and evaluated a text generation model using the Sentiment140 dataset. We used natural language processing techniques to preprocess the data, trained the model using the GPT-2 language model, and implemented a text generation algorithm to generate new tweets. We also evaluated the performance of the model using automatic evaluation metrics and human evaluation. Overall, our model achieved good performance in terms of perplexity, BLEU score, and ROUGE score, and received positive feedback from human evaluators.

10. REFERENCE

1. https://aclanthology.org/2022.findings-emnlp.509.pdf
2. file:///C:/Users/pragy/Downloads/2201.05337.pdf
3. https://arxiv.org/ftp/arxiv/papers/1905/1905.01984.pdf
4. https://arxiv.org/ftp/arxiv/papers/1905/1905.01984.pdf