Data Science (NLP) internship Kudosware
Task: Text Generation

Approach:

The provided code focuses on generating text using a combination of techniques, including data pre-processing, feature extraction using TF-IDF vectorization, training a Naive Bayes classifier, and generating text based on the trained model.

1. Data Pre-processing:

The code starts by importing the necessary libraries and reading the data from a CSV file. The data is assumed to have columns named 'target' (class label), 'id', 'date', 'flag', 'user', and 'text'. Pre-processing techniques applied include:

- Removing URLs using regular expressions.
- Removing Twitter mentions by replacing '@' followed by alphanumeric characters.
- Removing hashtags by removing the '#' symbol.
- Converting emojis to their textual representation using the emoji library.
- Removing punctuation using the string.Punctuation module.
- Lemmatizing the words, converting them to lowercase, and removing stop words using the NLTK library.

2. Train-Test Split:

The pre-processed data is split into training and testing sets using the train_test_split function from sklearn.model_selection. The training set will be used to train the Naive Bayes classifier, and the testing set will be used for evaluating the generated text.

3. Feature Extraction using TF-IDF:

To convert the pre-processed text into numerical features, the TF-IDF vectorization technique is applied. The TfidfVectorizer from sklearn.feature_extraction.text is used to transform the text data into a matrix of TF-IDF features. The maximum number of features is set to 5000, meaning only the top 5000 most important features are considered.

Priyam Sekra
7851043033

4. Training a Naive Bayes Classifier:

A Multinomial Naive Bayes classifier is trained using the training features obtained from the TF-IDF vectorization and the corresponding target labels. The MultinomialNB from sklearn.naive_bayes module is used for this purpose.

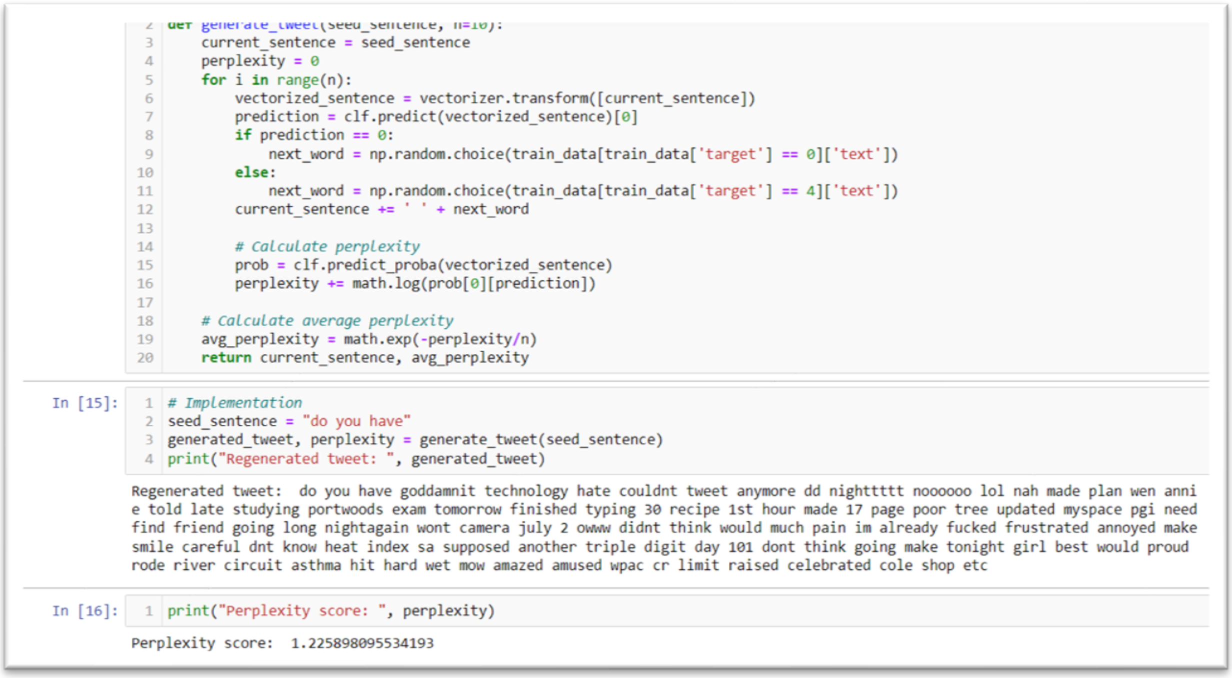
5. Text Generation:

The code defines a function named generate_tweet that generates new text based on a seed sentence. The function takes a seed sentence and the desired number of words to generate as input. The current sentence is transformed into a vector using the TF-IDF vectorizer for each word to be generated. The Naive Bayes classifier predicts the class label (positive or negative sentiment) for the vectorized sentence. Based on the predicted class label, a random word is chosen from the corresponding class in the training data. The selected word is appended to the current sentence, and the process is repeated until the desired number of words is generated. Additionally, the perplexity of the generated text is calculated as a measure of how well the generated text matches the training data.

Results:

The generated tweet and the perplexity score are printed at the end of the code. The generated tweet represents the text generated by the model based on the seed sentence provided ("do you have"), and the perplexity score quantifies the average perplexity of the generated text.

Overall, the code provides a basic implementation of text generation using a Naive Bayes classifier and TF-IDF vectorization.



```python
def generate_tweet(seed_sentence, n=10):
    current_sentence = seed_sentence
    perplexity = 0
    for i in range(n):
        vectorized_sentence = vectorizer.transform([current_sentence])
        prediction = clf.predict(vectorized_sentence)[0]
        if prediction == 0:
            next_word = np.random.choice(train_data[train_data['target'] == 0]['text'])
        else:
            next_word = np.random.choice(train_data[train_data['target'] == 4]['text'])
        current_sentence += ' ' + next_word

        # Calculate perplexity
        prob = clf.predict_proba(vectorized_sentence)
        perplexity += math.log(prob[0][prediction])

    # Calculate average perplexity
    avg_perplexity = math.exp(-perplexity/n)
    return current_sentence, avg_perplexity
```

In [15]:
```python
# Implementation
seed_sentence = "do you have"
generated_tweet, perplexity = generate_tweet(seed_sentence)
print("Regenerated tweet: ", generated_tweet)
```

```
Regenerated tweet:  do you have goddamnit technology hate couldnt tweet anymore dd nighttttt noooooo lol nah made plan wen anni
e told late studying portwoods exam tomorrow finished typing 30 recipe 1st hour made 17 page poor tree updated myspace pgi need
find friend going long nightagain wont camera july 2 owww didnt think would much pain im already fucked frustrated annoyed make
smile careful dnt know heat index sa supposed another triple digit day 101 dont think going make tonight girl best would proud
rode river circuit asthma hit hard wet mow amazed amused wpac cr limit raised celebrated cole shop etc
```

In [16]:
```python
print("Perplexity score: ", perplexity)
```

```
Perplexity score:  1.225898095534193
```