CS156 Final Pipeline

April 20, 2024

1 Introduction

This third pipeline is a continuation of my first and second pipeline with the same dataset, my personal online journal entries. For a quick recap, I did a sentiment analysis on my journal entries in my first pipeline to explore how much more positive the writings became with time. They didn't. But I also ended up creating a model that could then predict the sentiment of any new journal entries I write, which was way more accurate than it should be. In my second pipeline, I worked on pre-processing my data a bit differently with data augmentation to increase my data size, and trying out new models with XGBoost and RandomForest.

In this third pipeline, I want to explore using RNNs to generate text in my own voice with the given data.

2 Data Pre-processing

This Pre-processing part is from my first and second pipeline, where I: 1. Ensure there is no non-English language in the data 2. Augment the data after tokenization using BERT to increase my dataset size

[2]:
```
%pip install langdetect
```

```
Collecting langdetect
  Downloading langdetect-1.0.9.tar.gz (981 kB)
                             981.5/981.5
kB 7.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from langdetect) (1.16.0)
Building wheels for collected packages: langdetect
  Building wheel for langdetect (setup.py) … done
  Created wheel for langdetect: filename=langdetect-1.0.9-py3-none-any.whl
size=993227
sha256=ac237cf36d55716a95ca3967b74133452a00b197a663a8207e65198c082d8ab3
  Stored in directory: /root/.cache/pip/wheels/95/03/7d/59ea870c70ce4e5a370638b5
462a7711ab78fba2f655d05106
Successfully built langdetect
Installing collected packages: langdetect
Successfully installed langdetect-1.0.9
```

[3]:
```
from langdetect import detect

def is_english(text):
    try:
        return detect(text) == 'en'
    except:
        return False
```

[4]:
```
# for images later
from IPython.display import Image
from IPython.core.display import HTML

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from bs4 import BeautifulSoup

# Open the HTML file
with open('posts_1.html', 'r') as file:
    html_content = file.read()

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(html_content, 'html.parser')

# Extract timestamp
# all the timestamps are in a div called '_3-94 _a6-o'
timestamps = []
for time in soup.find_all(class_='_3-94 _a6-o'):
    timestamps.append(time.text.strip())


# Extract entry text
# all the entries are in a div called '_3-95 _2pim _a6-h _a6-i'
# why are all the div class names so weird?
entries = []
for entry in soup.find_all(class_='_3-95 _2pim _a6-h _a6-i'):
    entries.append(entry.text.strip())


# okay so now I can zip these 2 lists together!
# assuming that it is all in the same order.
date_entry = list(zip(timestamps, entries))
date_entry[0]

# re-ordering the list from the beginning, earliest entry
sorted_date_entry = date_entry[::-1]
```

```
sorted_date_entry[0]

# I just create a dataframe using the list above!
df = pd.DataFrame(sorted_date_entry, columns=['Date', 'Entry'])

# Filter out non-English entries
df = df[df['Entry'].apply(is_english)]

df_eng = df.drop(43)
df_eng
```

[4]:

	Date	Entry
0	27 Jun 2021, 17:30	i miss song river's roasted chicken drumsticks.
1	29 Jun 2021, 00:51	so many nice memories from penang :>
2	29 Jun 2021, 01:06	i think thats the first braid i ever did corre…
3	30 Jun 2021, 01:26	horhorhor cant wait to leave this place
4	1 Jul 2021, 02:39	seobseob allthetiem
..	…	…
124	3 Sep 2023, 06:53	i hate this feeling of everything starting to …
126	18 Sep 2023, 18:06	we completely broke up yesterday.\ni miss him …
127	28 Sep 2023, 16:55	i miss him so much how did i even get over rya…
128	6 Jan 2024, 10:05	just to update.\nmy BA semester:\ni arrived th…
129	9 Jan 2024, 16:47	the british museum, malatang, snow, WICKED and…

[109 rows x 2 columns]

[5]:
```
%pip install nlpaug
%pip install nltk

%pip install gensim
%pip install transformers
```

```
Collecting nlpaug
  Downloading nlpaug-1.1.11-py3-none-any.whl (410 kB)
                           410.5/410.5
kB 6.3 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.16.2 in
/usr/local/lib/python3.10/dist-packages (from nlpaug) (1.25.2)
Requirement already satisfied: pandas>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from nlpaug) (2.0.3)
Requirement already satisfied: requests>=2.22.0 in
/usr/local/lib/python3.10/dist-packages (from nlpaug) (2.31.0)
Requirement already satisfied: gdown>=4.0.0 in /usr/local/lib/python3.10/dist-
packages (from nlpaug) (4.7.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from gdown>=4.0.0->nlpaug) (3.13.4)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
```

```
(from gdown>=4.0.0->nlpaug) (1.16.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from gdown>=4.0.0->nlpaug) (4.66.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from gdown>=4.0.0->nlpaug) (4.12.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.2.0->nlpaug) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2.0->nlpaug) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2.0->nlpaug) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0->nlpaug) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.22.0->nlpaug) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0->nlpaug) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0->nlpaug)
(2024.2.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->gdown>=4.0.0->nlpaug) (2.5)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0->nlpaug) (1.7.1)
Installing collected packages: nlpaug
Successfully installed nlpaug-1.1.11
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.4.0)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.2)
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages
(4.3.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-
packages (from gensim) (1.25.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-
packages (from gensim) (1.11.4)
Requirement already satisfied: smart-open>=1.8.1 in
/usr/local/lib/python3.10/dist-packages (from gensim) (6.4.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.38.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.13.4)
```

```
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.15.2)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.2)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.19.3->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.19.3->transformers) (4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.2.2)
```

[6]:
```python
import torch
from transformers import BertTokenizer, BertModel
import pandas as pd
import nlpaug.augmenter.word as naw

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Define a function to generate BERT embeddings for a given text
def get_bert_embeddings(text):
    # Tokenize input text
    inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True)
```

```
    # Forward pass through BERT model
    with torch.no_grad():
        outputs = model(**inputs)

    # Extract BERT embeddings (output of the last hidden layer)
    embeddings = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()

    return embeddings

# Initialize DataFrame to store augmented entries
df_augmented = pd.DataFrame()

# Initialize original entries
original_entries = df_eng['Entry'].tolist()

# Define the number of iterations
num_iterations = 5

# Initialize augmenter
aug = naw.ContextualWordEmbsAug(model_path='bert-base-uncased', action="insert")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
```

```
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
```

```
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

```
model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]
```

[7]:
```
# Augment entries recursively
for i in range(num_iterations):
    # Initialize lists to store augmented entries and their embeddings
    augmented_entries = []
    bert_embeddings = []

    # Augment each entry separately
    for entry in original_entries:
```

```
        # Augment entry
        augmented_entry = aug.augment(entry)
        augmented_entry_str = ' '.join(augmented_entry)
        augmented_entries.append(augmented_entry_str)

        # Generate BERT embeddings for augmented entry
        embeddings = get_bert_embeddings(augmented_entry_str)
        bert_embeddings.append(embeddings)

    # Add augmented entries and embeddings to DataFrame
    df_iteration = pd.DataFrame({'Entry': original_entries,
                                 'Augmented Entry': augmented_entries,
                                 'BERT Embeddings': bert_embeddings})

    # Append DataFrame for this iteration to the main DataFrame
    #df_augmented = df_augmented.concat(df_iteration, ignore_index=True)
    df_augmented = pd.concat([df_augmented, df_iteration], ignore_index=True)

    # Update original entries for the next iteration
    original_entries = augmented_entries

# Print DataFrame with augmented entries
df_augmented
```

[7]:

```
                                                 Entry  \
0      i miss song river's roasted chicken drumsticks.
1                  so many nice memories from penang :>
2    i think thats the first braid i ever did corre…
3              horhorhor cant wait to leave this place
4                                 seobseob allthetiem
..                                                 …
540  i hate this feeling of everything starting to …
541  when we id still completely broke up yesterday…
542  i miss him so very damn to very much how early…
543  oh just to update. my ba semester : thursday i…
544  just around nearby entrance at nearby the then…

                                      Augmented Entry  \
0    yes i miss song in river'fish s roasted dried …
1    more so how many nice memories come from penan…
2    i just think thats the first braid that i ever…
3    horhorhor cant no wait ever to really leave to…
4                            a seobseob no allthetiem
..                                                 …
540  i hate this feeling of everything starting to …
541  when we id still completely broke up yesterday…
542  i miss this him so very damn much to very much…
```

```
543  oh just to update. my ba semester : thursday i…
544  just around nearby entrance at nearby the then…

                                          BERT Embeddings
0    [0.15325797, 0.022680618, 0.562708, -0.1315718…
1    [0.27629235, 0.012392225, 0.64009374, -0.17464…
2    [-0.18537118, 0.2270245, 0.20495255, 0.1505301…
3    [-0.13521999, 0.24002987, 0.06469188, -0.26303…
4    [-0.38230804, 0.13267045, -0.13536943, -0.2708…
..                                                 …
540  [-0.18831538, 0.19258283, 0.38677037, -0.03363…
541  [-0.15145852, 0.08331992, 0.47926685, 0.109926…
542  [-0.13092694, 0.35802293, 0.67328286, -0.09456…
543  [-0.13224922, 0.020649241, 0.5754887, 0.072461…
544  [-0.22872409, 0.23886026, 0.40510458, -0.18309…

[545 rows x 3 columns]
```

[8]:
```
#This code is adapted from Datacamp: https://www.datacamp.com/tutorial/
 ↪text-analytics-beginners-nltk
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package vader_lexicon to /root/nltk_data…
```

[9]:
```
# create preprocess_text function
def preprocess_text(text):

    # Tokenize the text
    tokens = word_tokenize(text.lower())


    # Remove stop words
    filtered_tokens = [token for token in tokens if token not in stopwords.
 ↪words('english')]
```

```
    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in␣
 ↪filtered_tokens]

    # Join the tokens back into a string
    processed_text = ' '.join(lemmatized_tokens)

    return processed_text

# apply the function to my df
print(df_augmented['Entry'])
df_augmented['Entry Processed'] = df_augmented['Entry'].apply(preprocess_text)
df_augmented['Augmented Entry Processed'] = df_augmented['Augmented Entry'].
 ↪apply(preprocess_text)
df_augmented
```

```
0        i miss song river's roasted chicken drumsticks.
1                    so many nice memories from penang :>
2      i think thats the first braid i ever did corre…
3                horhorhor cant wait to leave this place
4                                    seobseob allthetiem
                              …
540    i hate this feeling of everything starting to …
541    when we id still completely broke up yesterday…
542    i miss him so very damn to very much how early…
543    oh just to update. my ba semester : thursday i…
544    just around nearby entrance at nearby the then…
Name: Entry, Length: 545, dtype: object
```

[9]:

```
                                                 Entry  \
0       i miss song river's roasted chicken drumsticks.
1                   so many nice memories from penang :>
2      i think thats the first braid i ever did corre…
3                horhorhor cant wait to leave this place
4                                    seobseob allthetiem
..                                                   …
540    i hate this feeling of everything starting to …
541    when we id still completely broke up yesterday…
542    i miss him so very damn to very much how early…
543    oh just to update. my ba semester : thursday i…
544    just around nearby entrance at nearby the then…

                                       Augmented Entry  \
0      yes i miss song in river'fish s roasted dried …
```

```
1    more so how many nice memories come from penan…
2    i just think thats the first braid that i ever…
3    horhorhor cant no wait ever to really leave to…
4                              a seobseob no allthetiem
..                                                  …
540  i hate this feeling of everything starting to …
541  when we id still completely broke up yesterday…
542  i miss this him so very damn much to very much…
543  oh just to update. my ba semester : thursday i…
544  just around nearby entrance at nearby the then…

                                        BERT Embeddings  \
0    [0.15325797, 0.022680618, 0.562708, -0.1315718…
1    [0.27629235, 0.012392225, 0.64009374, -0.17464…
2    [-0.18537118, 0.2270245, 0.20495255, 0.1505301…
3    [-0.13521999, 0.24002987, 0.06469188, -0.26303…
4    [-0.38230804, 0.13267045, -0.13536943, -0.2708…
..                                                  …
540  [-0.18831538, 0.19258283, 0.38677037, -0.03363…
541  [-0.15145852, 0.08331992, 0.47926685, 0.109926…
542  [-0.13092694, 0.35802293, 0.67328286, -0.09456…
543  [-0.13224922, 0.020649241, 0.5754887, 0.072461…
544  [-0.22872409, 0.23886026, 0.40510458, -0.18309…

                                        Entry Processed  \
0       miss song river 's roasted chicken drumstick .
1                            many nice memory penang : >
2                think thats first braid ever correctly
3                       horhorhor cant wait leave place
4                                    seobseob allthetiem
..                                                  …
540  hate feeling everything starting feel like cho…
541  id still completely broke yesterday . like mis…
542  miss damn much early even get ryan . ok ok wel…
543  oh update . ba semester : thursday recently ar…
544  around nearby entrance nearby modern national …

                               Augmented Entry Processed
0    yes miss song river'fish roasted dried chicken…
1                      many nice memory come penang : >
2           think thats first braid ever ever correctly
3     horhorhor cant wait ever really leave town place
4                                    seobseob allthetiem
..                                                  …
540  hate feeling everything starting feel like cho…
541  id still completely broke yesterday . like mis…
542  miss damn much much damn early even get saying…
```

```
543  oh update . ba semester : thursday recently ar…
544  around nearby entrance nearby demolished moder…

[545 rows x 5 columns]
```

[10]:
```
# initialize NLTK sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# create get_scores function
def get_scores(text):
    return analyzer.polarity_scores(text)['compound']

# create get_sentiment function
def get_sentiment(text):
    scores = get_scores(text)
    if scores > 0:
        sentiment = 'Positive'
    elif scores == 0:
        sentiment = 'Neutral'
    else:
        sentiment = 'Negative'

    return sentiment


df_augmented['Scores'] = df_augmented['Entry'].apply(get_scores)
df_augmented['Sentiment'] = df_augmented['Entry'].apply(get_sentiment)
df_augmented['Augmented Sentiment'] = df_augmented['Augmented Entry Processed'].
 ↪apply(get_sentiment)

df_augmented['Scores Processed'] = df_augmented['Entry Processed'].
 ↪apply(get_scores)
df_augmented['Sentiment Processed'] = df_augmented['Entry Processed'].
 ↪apply(get_sentiment)

df_augmented
```

[10]:
```
                                                Entry  \
0      i miss song river's roasted chicken drumsticks.
1                  so many nice memories from penang :>
2    i think thats the first braid i ever did corre…
3              horhorhor cant wait to leave this place
4                                  seobseob allthetiem
..                                                 …
540  i hate this feeling of everything starting to …
541  when we id still completely broke up yesterday…
542  i miss him so very damn to very much how early…
```

```
543  oh just to update. my ba semester : thursday i…
544  just around nearby entrance at nearby the then…

                                      Augmented Entry  \
0    yes i miss song in river'fish s roasted dried …
1    more so how many nice memories come from penan…
2    i just think thats the first braid that i ever…
3    horhorhor cant no wait ever to really leave to…
4                              a seobseob no allthetiem
..                                                   …
540  i hate this feeling of everything starting to …
541  when we id still completely broke up yesterday…
542  i miss this him so very damn much to very much…
543  oh just to update. my ba semester : thursday i…
544  just around nearby entrance at nearby the then…

                                      BERT Embeddings  \
0    [0.15325797, 0.022680618, 0.562708, -0.1315718…
1    [0.27629235, 0.012392225, 0.64009374, -0.17464…
2    [-0.18537118, 0.2270245, 0.20495255, 0.1505301…
3    [-0.13521999, 0.24002987, 0.06469188, -0.26303…
4    [-0.38230804, 0.13267045, -0.13536943, -0.2708…
..                                                   …
540  [-0.18831538, 0.19258283, 0.38677037, -0.03363…
541  [-0.15145852, 0.08331992, 0.47926685, 0.109926…
542  [-0.13092694, 0.35802293, 0.67328286, -0.09456…
543  [-0.13224922, 0.020649241, 0.5754887, 0.072461…
544  [-0.22872409, 0.23886026, 0.40510458, -0.18309…

                                      Entry Processed  \
0       miss song river 's roasted chicken drumstick .
1                           many nice memory penang : >
2                think thats first braid ever correctly
3                       horhorhor cant wait leave place
4                                  seobseob allthetiem
..                                                   …
540  hate feeling everything starting feel like cho…
541  id still completely broke yesterday . like mis…
542  miss damn much early even get ryan . ok ok wel…
543  oh update . ba semester : thursday recently ar…
544  around nearby entrance nearby modern national …

                              Augmented Entry Processed  Scores Sentiment  \
0    yes miss song river'fish roasted dried chicken… -0.1531  Negative
1                      many nice memory come penang : >  0.7334  Positive
2           think thats first braid ever ever correctly  0.0000   Neutral
3     horhorhor cant wait ever really leave town place  0.0382  Positive
```

```
4                                  seobseob allthetiem  0.0000   Neutral
..                                                    …       …       …
540  hate feeling everything starting feel like cho… -0.9253  Negative
541  id still completely broke yesterday . like mis…  0.9869  Positive
542  miss damn much much damn early even get saying… -0.1613  Negative
543  oh update . ba semester : thursday recently ar… -0.9608  Negative
544  around nearby entrance nearby demolished moder…  0.1779  Positive

    Augmented Sentiment  Scores Processed Sentiment Processed
0              Positive           -0.1531             Negative
1              Positive            0.4215             Positive
2               Neutral            0.0000              Neutral
3              Negative            0.0382             Positive
4               Neutral            0.0000              Neutral
..                  …                 …                 …
540            Negative           -0.8457             Negative
541            Positive            0.9736             Positive
542            Negative           -0.1346             Negative
543            Positive            0.6075             Positive
544            Positive            0.1779             Positive

[545 rows x 10 columns]
```

[11]:
```
import xgboost
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score

# This is from first pipeline:
# first I split the dataframe into training and testing datasets
X = df_augmented['Augmented Entry Processed']
y = df_augmented['Augmented Sentiment']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# I then vectorize the text data
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Encode the categorical target variable into numerical values
```

```
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)

# I have 3 different models I want to compare
model1 = XGBClassifier(n_estimators=100, max_depth=3, random_state=42) #XGBoost
model2 = RandomForestClassifier(n_estimators = 100, random_state = 42) #␣
 ↪RandomForest
model3 = MLPClassifier(hidden_layer_sizes=(100,),
                       max_iter=300, activation='relu',
                       solver='adam', random_state=42) # SVM

# now all the models undergo training
model1.fit(X_train_tfidf, y_train_encoded)
model2.fit(X_train_tfidf, y_train)
model3.fit(X_train_tfidf, y_train)


# and then I make predictions using the testing data, not the
# training data, to avoid in-sample scores.
y_pred1 = model1.predict(X_test_tfidf)
y_pred2 = model2.predict(X_test_tfidf)
y_pred3 = model3.predict(X_test_tfidf)

y_pred1 = label_encoder.inverse_transform(y_pred1)


# I use sklearn's accuracy_score function to calculate the accuracy
# and use it as a metric to compare the models
accuracy1 = accuracy_score(y_test, y_pred1)
accuracy2 = accuracy_score(y_test, y_pred2)
accuracy3 = accuracy_score(y_test, y_pred3)
print("Accuracy with XGBoost:", accuracy1)
print("Accuracy with Random Forest:", accuracy2)
print("Accuracy with SVM:", accuracy3, '\n')

# I then test my models on new data
new_text = ["i am so sad", "im really happy!"]
actual = ['Negative', 'Positive']

new_text_preprocessed = pd.Series(new_text).apply(preprocess_text)
new_text_tfidf = tfidf_vectorizer.transform(new_text_preprocessed)
predictions1 = model1.predict(new_text_tfidf)
pred1_decode = label_encoder.inverse_transform(predictions1) # to decode the␣
 ↪integers to classifiers
predictions2 = model2.predict(new_text_tfidf)
predictions3 = model3.predict(new_text_tfidf)
```

```
print(f"New test text: {new_text}")
print(f"Predictions with XGBoost: {pred1_decode}" )
print(f"Predictions with Random Forest: {predictions2}")
print(f"Predictions with SVM: {predictions3}")
```

```
Accuracy with XGBoost: 0.8807339449541285
Accuracy with Random Forest: 0.9174311926605505
Accuracy with SVM: 0.8899082568807339

New test text: ['i am so sad', 'im really happy!']
Predictions with XGBoost: ['Positive' 'Positive']
Predictions with Random Forest: ['Neutral' 'Positive']
Predictions with SVM: ['Negative' 'Positive']
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:686:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

2.0.1 Random forest

A Random Forest classifier is an ensemble learner, meaning that it uses the output of many decision trees to lead to a final leaf node. Then the most 'popular' outcome in an ensemble of trees is aggregated, usually by averaging them together (Hossein Ashtari, 2024).

But as seen from the accuracy results above, it seems like the best model numerically is XGBoost:

2.0.2 XGBoost

XGBoost is a supervised learning method that minimizes the following objective function: $^{t} = \overset{n}{i=1} l(y_i, \vec{y}_i^{t-1} + f_t(x_i)) + (f_t)$ Where: Of course, let's break down each term in the objective function:

- ^{t}: this represents the objective function that XGBoost aims to minimize at each iteration t of the boosting process.
- $l(y_i, \vec{y}_i^{t-1} + f_t(x_i))$: this term measures the discrepancy between the actual target value y_i and the prediction made by the ensemble up to iteration $t-1$ plus the prediction of the current weak learner $f_t(x_i)$). It quantifies how much the current weak learner improves the model's prediction compared to the existing ensemble.
- (f_t): this term penalizes the complexity of the current weak learner $f_t(x_i)$. It discourages overly complex models by adding a penalty term to the objective function. Common regularization techniques include constraints on tree depth, leaf weights, and leaf node counts.

The loss function is the first term, l, which measures how far the prediction is from the target value. Then the second term, Ω, is the regularization function, which penalizes the current weak learner in the model, ensuring that there is no overfitting. This objective function is minimized at each iteration of t timestep for each subtree.

Basically we carry out optimization on a convex loss function while penalizing any increasing complexity of the model.

By using the second-order Taylor approximation on l, the objecive function can be rewritten as: $^{t} = \sum_{i=1}^{n} l(y_i, \vec{y}_i^{t-1} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + (f_t)$

which after removing the constants, can simplify to: $^{t} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + (f_t)$

This basically becomes a quadratic expression of $f_t(x_i)$ that can be minimized at each step (Hossein Ashtari, 2024).

2.0.3 The best model?

Even though the accuracy score for XGBoost is high (like for the other 2 models), it seems to struggle on the new test text prediction. Out of the 3 models, it is the only one that fails to predict 'Negative' and 'Positive'. The accuracy score as a metric only measures how many properly classified texts my models have out of the split testing data.

It may even be the case that my models are just too heavily trained on my data, so it overfit to text only from my entries.

SVM, as shown in the first pipeline, is really good at handling class imabalances, whereas XGBoost (which I had a lot of hope for, since it is a model I commonly see in many Kaggle notebooks) only works on optimizing an objective function. So it does not explicitly handle any imbalance in the classes in the given dataset. Since my dataset has a lot more positive tags (only 19.3% of my data is negative, from my previous assignment), XGBoost may be more used to looking for positive tags.

Hence, I conclude that with my limited and flawed dataset, **SVM is the best model**.

3 Data Processing

This part is done by following this tutorial: Text generation with an RNN. (2023). TensorFlow. https://www.tensorflow.org/text/tutorials/text_generation

As done in previous pipelines, we need to first vectorize the text so that the computer can understand and manipulate them more easily. Simply put, we need to convert the strings into numbers. To do this, we first get all the entries from our pre-processed data:

[12]:
```
import tensorflow as tf
import os
import time

# Concatenate all sentences into a single string
all_text = ' '.join(df_augmented['Entry'])

# Tokenize the text into characters
chars = list(set(all_text.lower()))

# Get unique vocabulary
vocab = sorted(chars)

# Print the number of unique characters and total length of the text
print(f'{len(vocab)} unique characters')
print(f'Length of text: {len(all_text)} characters')
```

```
99 unique characters
Length of text: 271984 characters
```

We then convert all of the texts into its own token (so each 'char'), which can be converted into ids. Since this model's output needs to be readable English text, not numbers, we then convert all the ids back into the tokens, and words.

[13]:
```
# convert the words into tokens to ids
chars = tf.strings.unicode_split(all_text, input_encoding='UTF-8')
ids_from_chars = tf.keras.layers.StringLookup(vocabulary=list(vocab),␣
 ↪mask_token=None)
ids = ids_from_chars(chars)

# convert ids back to tokens
chars_from_ids = tf.keras.layers.StringLookup(vocabulary=ids_from_chars.
 ↪get_vocabulary(), invert=True, mask_token=None)
chars = chars_from_ids(ids)

tf.strings.reduce_join(chars, axis=-1).numpy()

# then convert the tokens into long strings joined together
def text_from_ids(ids):
  return tf.strings.reduce_join(chars_from_ids(ids), axis=-1)
```

Now we convert all of the ids into one vector, and use the batch method to group together all the ids into our desired seq_length. A longer sequence length can capture more context but is more computationally expensive to run. So I start with 200. This means that every 200 tokens will be considered as the input, and the target will be the next token from the original text.

[14]:
```
all_ids = ids_from_chars(tf.strings.unicode_split(all_text, 'UTF-8'))

# convert ids into a vector of ids
ids_dataset = tf.data.Dataset.from_tensor_slices(all_ids)

# to check if we can decode!
# for ids in ids_dataset.take(10):
#     print(chars_from_ids(ids).numpy().decode('utf-8'))

seq_length = 200

sequences = ids_dataset.batch(seq_length+1, drop_remainder=True)

for seq in sequences.take(1):
  print(chars_from_ids(seq))

# tokens back into strings
for seq in sequences.take(5):
  print(text_from_ids(seq).numpy())
```

```python
def split_input_target(sequence):
  input_text = sequence[:-1]
  target_text = sequence[1:]
  return input_text, target_text

dataset = sequences.map(split_input_target)
for input_example, target_example in dataset.take(1):
    print("Input :", text_from_ids(input_example).numpy())
    print("Target:", text_from_ids(target_example).numpy())
```

```
tf.Tensor(
[b'i' b' ' b'm' b'i' b's' b's' b' ' b's' b'o' b'n' b'g' b' ' b'r' b'i'
 b'v' b'e' b'r' b"'" b's' b' ' b'r' b'o' b'a' b's' b't' b'e' b'd' b' '
 b'c' b'h' b'i' b'c' b'k' b'e' b'n' b' ' b'd' b'r' b'u' b'm' b's' b't'
 b'i' b'c' b'k' b's' b'.' b' ' b's' b'o' b' ' b'm' b'a' b'n' b'y' b' '
 b'n' b'i' b'c' b'e' b' ' b'm' b'e' b'm' b'o' b'r' b'i' b'e' b's' b' '
 b'f' b'r' b'o' b'm' b' ' b'p' b'e' b'n' b'a' b'n' b'g' b' ' b':' b'>'
 b' ' b'i' b' ' b't' b'h' b'i' b'n' b'k' b' ' b't' b'h' b'a' b't' b's'
 b' ' b't' b'h' b'e' b' ' b'f' b'i' b'r' b's' b't' b' ' b'b' b'r' b'a'
 b'i' b'd' b' ' b'i' b' ' b'e' b'v' b'e' b'r' b' ' b'd' b'i' b'd' b' '
 b'c' b'o' b'r' b'r' b'e' b'c' b't' b'l' b'y' b' ' b'h' b'o' b'r' b'h'
 b'o' b'r' b'h' b'o' b'r' b' ' b'c' b'a' b'n' b't' b' ' b'w' b'a' b'i'
 b't' b' ' b't' b'o' b' ' b'l' b'e' b'a' b'v' b'e' b' ' b't' b'h' b'i'
 b's' b' ' b'p' b'l' b'a' b'c' b'e' b' ' b's' b'e' b'o' b'b' b's' b'e'
 b'o' b'b' b' ' b'a' b'l' b'l' b't' b'h' b'e' b't' b'i' b'e' b'm' b' '
 b'i' b' ' b'w' b'a' b'n'], shape=(201,), dtype=string)
b"i miss song river's roasted chicken drumsticks. so many nice memories from
penang :> i think thats the first braid i ever did correctly horhorhor cant wait
to leave this place seobseob allthetiem i wan"
b't to fucking cry. really great memories. he would have found these cool. lol
results come out in 8 hours and his birthday is in 12. i deserve it lol. im so
stupid and now everyone knows. probably a goo'
b'd thing it ended too. i get scared from every new email incase its minerva
rescinding me lol. i cant get over him and am anxious af waiting for my uni to
reply so here are a list of things i wanna do o'
b'ne day when i fall back in love.\n\n1. back hugs while swaying to music in a
dim room\n2. forehead and nose kisses\n3. eating midnight snacks together\n4.
dancing in the kitchen while cooking together\n5. do'
b'ing a movie marathon of some sort together (httyd trilogy, or harry potter?
idk)\n6. going skiing together\n7. double or even triple dates!\n8. vacation to
hawaii together\n9. watch the sunset/sunrise toge'
Input : b"i miss song river's roasted chicken drumsticks. so many nice memories
from penang :> i think thats the first braid i ever did correctly horhorhor cant
wait to leave this place seobseob allthetiem i wa"
Target: b" miss song river's roasted chicken drumsticks. so many nice memories
from penang :> i think thats the first braid i ever did correctly horhorhor cant
wait to leave this place seobseob allthetiem i wan"
```

Then we create training batches, which are like subsets of our training dataset. This helps to optimize the parameters (like weights and biases) during training with each new batch of data. More batches helps to increase the generalizability of the model's outputs, but can be computationally expensive. 64 is a common batch size so I leave it as 64.

[15]:
```
# Batch size
BATCH_SIZE = 64

# Buffer size to shuffle the dataset
# (TF data is designed to work with possibly infinite sequences,
# so it doesn't attempt to shuffle the entire sequence in memory. Instead,
# it maintains a buffer in which it shuffles elements).
BUFFER_SIZE = 10000

dataset = (
    dataset
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE, drop_remainder=True)
    .prefetch(tf.data.experimental.AUTOTUNE))

dataset
```

[15]:
```
<_PrefetchDataset element_spec=(TensorSpec(shape=(64, 200), dtype=tf.int64,
name=None), TensorSpec(shape=(64, 200), dtype=tf.int64, name=None))>
```

4 Building the RNN

https://www.tensorflow.org/text/tutorials/text_generation

[1]:
```
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://i.imgur.com/RhH4cpk.png", width = 500)
```

[1]:
```
<IPython.core.display.Image object>
```

We finally start building our model after all that processing! I follow the Tensorflow tutorial online on building an RNN from scratch.

An RNN is a class of neural networks that can handle sequential data. By maintaining a memory of previous inputs, RNNs can capture temporal dependencies and learn from the context of past inputs, making them well-suited for tasks like natural language processing, time series prediction, and sequence generation. So it similarlity uses back propogation techniques like we saw in Feed-foward neural networks to update the gradient of the loss function with respect to the parameters at each time step during training„ so that it can minimize the loss function (like cross-entropy)

So there is an input layer, hidden layer, and output layer. In the hidden layer are many different hidden states. The hidden state (h_t) at time step (t) is calculated as follows:

$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$

Where: - (W_{hx}) is the weight matrix connecting the input to the hidden state. - $(W_{hh}) $ is the weight matrix connecting the previous hidden state to the current hidden state. - (b_h) is the bias vector. - (f) is the activation function applied element-wise to the sum of the weighted inputs and bias.

The output (y_t) at time step (t) is calculated based on the current hidden state: $ y_t = g(W_{yh}h_t + b_y) $

- (W_{yh}) is the weight matrix connecting the hidden state to the output.
- (b_y) is the bias vector.
- (g) is the activation function applied element-wise to the sum of the weighted hidden state and bias.

[16]:
```
# Length of the vocabulary in StringLookup Layer
vocab_size = len(ids_from_chars.get_vocabulary())

# The embedding dimension
embedding_dim = 256

# Number of RNN units
rnn_units = 1024
```

[17]:
```
class MyModel(tf.keras.Model):
  def __init__(self, vocab_size, embedding_dim, rnn_units):
    super().__init__(self)
    self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
    self.gru = tf.keras.layers.GRU(rnn_units,
                                   return_sequences=True,
                                   return_state=True)
    self.dense = tf.keras.layers.Dense(vocab_size)

  def call(self, inputs, states=None, return_state=False, training=False):
    x = inputs
    x = self.embedding(x, training=training)
    if states is None:
      states = self.gru.get_initial_state(x)
    x, states = self.gru(x, initial_state=states, training=training)
    x = self.dense(x, training=training)

    if return_state:
      return x, states
    else:
      return x
```

[18]:
```
model = MyModel(
    vocab_size=vocab_size,
    embedding_dim=embedding_dim,
    rnn_units=rnn_units)
```

```
for input_example_batch, target_example_batch in dataset.take(1):
    example_batch_predictions = model(input_example_batch)
    print(example_batch_predictions.shape, "# (batch_size, sequence_length,␣
 ↪vocab_size)")

model.summary()
```

```
(64, 200, 100) # (batch_size, sequence_length, vocab_size)
Model: "my_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       multiple                  25600

 gru (GRU)                   multiple                  3938304

 dense (Dense)               multiple                  102500

=================================================================
Total params: 4066404 (15.51 MB)
Trainable params: 4066404 (15.51 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[19]:
```
sampled_indices = tf.random.categorical(example_batch_predictions[0],␣
 ↪num_samples=1)
sampled_indices = tf.squeeze(sampled_indices, axis=-1).numpy()

print("Input:\n", text_from_ids(input_example_batch[0]).numpy())
print()
print("Next Char Predictions:\n", text_from_ids(sampled_indices).numpy())
```

```
Input:
 b'anger than someone that u so kinda must know. cuz like maybe the stranger u
just will never see again + they dont know u or the people in ur own life so in
a sense u can trust them to not tell anyone.'

Next Char Predictions:
 b"\xe1\x86\xbc\xec\x84\xb1\xe1\x84\x86;eq\xe1\x84\x86kz\n-\xf0\x9f\x98\x974\xe2
\x80\x99\xed\x95\x98\xe2\x80\x8d21+r\xc2\xbdvm\xe2\x80\xb2i/\xe1\x84\x82l\xea\xb
5\xadir3\xed\x8a\xb9p\xed\x95\x98s\xe1\x86\xbc\xf0\x9f\xa4\xa9\xec\xbd\xa9:\xe1\
x85\xb5\xe1\x86\xa8)4kd\xe1\x84\x92;\xef\xb8\x8f\xf0\x9f\x98\x97\xed\x8a\xb9/\xc
2\xbd\xe1\x84\x8fa15\xf0\x9f\xa5\xba\xe1\x86\xbc' \xeb\x82\x98\xe1\x85\xb5'-\xea
\xb8\x89>\xed\x95\x98\xc2\xbbdn\xf0\x9f\x99\x88~\xe1\x85\xae4']+868y\xe2\x80\xb2
\xe1\x84\x92\xe1\x84\x92<?\xec\x84\xb1\xf0\x9f\x98\xac\xf0\x9f\x98\x97\xed\x8a\x
b9g\xe1\x84\x86?\xc2\xbb\xe1\x85\xa9va\xe1\x84\x86\xef\xb8\x8f\xec\x9d\xb4\xf0\x
9f\xa5\xb5:\xed\x95\x98w?\xe2\x80\x8d\xf0\x9f\x98\x8c\xf0\x9f\xa4\xa1xx#\xf0\x9f
```

```
\x98\x97\xe1\x84\x8b0\xc2\xbdd\xeb\x82\x98k e).\xc2\xab;]\xe1\x84\x92\xec\x84\xb
1\xe1\x86\xaf\xe1\x84\x86\xec\x9d\xb48~d8xo!v\xc2\xbd\xec\x84\xb1g\xf0\x9f\x91\x
84\xed\x98\x9ct\xea\xb8\x89\xc2\xab8\xf0\x9f\xa4\xa1\xec\x84\xb1x\xeb\x82\x98\xe
b\xac\xbc>51q[UNK]8;1\xf0\x9f\x91\x811 \xf0\x9f\x98\x8ci\xe1\x85\xb5\xf0\x9f\xa4
\xa1~\n\xec\x9d\xb4\xe1\x85\xaesqp\xe1\x85\xa9o\xec\xbd\xa9?\xe1\x86\xa8#6\xf0\x
9f\xa4\xa9\xf0\x9f\x98\xac2\xe1\x86\xafvp[9+#\xe1\x85\xb5\xe1\x84\x8ft46e\xe1\x8
5\xae\xed\x8a\xb9"
```

Attaching an optimizer and loss function

[20]:
```
loss = tf.losses.SparseCategoricalCrossentropy(from_logits=True)
example_batch_mean_loss = loss(target_example_batch, example_batch_predictions)
print("Prediction shape:", example_batch_predictions.shape, " # (batch_size,␣
 ↪sequence_length, vocab_size)")
print("Mean loss:", example_batch_mean_loss)
print("Exponential of mean loss:",tf.exp(example_batch_mean_loss).numpy())
# roughly similar to vocabulary size, as this model has not undergone training␣
 ↪yet!
```

```
Prediction shape: (64, 200, 100)  # (batch_size, sequence_length, vocab_size)
Mean loss: tf.Tensor(4.60399, shape=(), dtype=float32)
Exponential of mean loss: 99.88206
```

[21]:
```
model.compile(optimizer='adam', loss=loss)
```

[53]:
```
# Directory where the checkpoints will be saved
checkpoint_dir = './training_checkpoints'
# Name of the checkpoint files
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt_{epoch}")

checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_prefix,
    save_weights_only=True)

EPOCHS = 40

history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 1/40
21/21 [==============================] - 254s 12s/step - loss: 1.1364
Epoch 2/40
21/21 [==============================] - 236s 11s/step - loss: 1.0501
Epoch 3/40
21/21 [==============================] - 240s 11s/step - loss: 0.9600
Epoch 4/40
21/21 [==============================] - 239s 11s/step - loss: 0.8690
Epoch 5/40
21/21 [==============================] - 236s 11s/step - loss: 0.7782
Epoch 6/40
```

```
21/21 [==============================] - 236s 11s/step - loss: 0.6899
Epoch 7/40
21/21 [==============================] - 237s 11s/step - loss: 0.6059
Epoch 8/40
21/21 [==============================] - 236s 11s/step - loss: 0.5245
Epoch 9/40
21/21 [==============================] - 236s 11s/step - loss: 0.4527
Epoch 10/40
21/21 [==============================] - 232s 11s/step - loss: 0.3936
Epoch 11/40
21/21 [==============================] - 235s 11s/step - loss: 0.3420
Epoch 12/40
21/21 [==============================] - 236s 11s/step - loss: 0.2968
Epoch 13/40
21/21 [==============================] - 234s 11s/step - loss: 0.2562
Epoch 14/40
21/21 [==============================] - 234s 11s/step - loss: 0.2254
Epoch 15/40
21/21 [==============================] - 239s 11s/step - loss: 0.1989
Epoch 16/40
21/21 [==============================] - 238s 11s/step - loss: 0.1779
Epoch 17/40
21/21 [==============================] - 236s 11s/step - loss: 0.1596
Epoch 18/40
21/21 [==============================] - 236s 11s/step - loss: 0.1444
Epoch 19/40
21/21 [==============================] - 238s 11s/step - loss: 0.1334
Epoch 20/40
21/21 [==============================] - 236s 11s/step - loss: 0.1232
Epoch 21/40
21/21 [==============================] - 238s 11s/step - loss: 0.1148
Epoch 22/40
21/21 [==============================] - 238s 11s/step - loss: 0.1072
Epoch 23/40
21/21 [==============================] - 236s 11s/step - loss: 0.1015
Epoch 24/40
21/21 [==============================] - 235s 11s/step - loss: 0.0959
Epoch 25/40
21/21 [==============================] - 234s 11s/step - loss: 0.0913
Epoch 26/40
21/21 [==============================] - 235s 11s/step - loss: 0.0872
Epoch 27/40
21/21 [==============================] - 238s 11s/step - loss: 0.0831
Epoch 28/40
21/21 [==============================] - 240s 11s/step - loss: 0.0805
Epoch 29/40
21/21 [==============================] - 238s 11s/step - loss: 0.0781
Epoch 30/40
```

```
21/21 [==============================] - 237s 11s/step - loss: 0.0756
Epoch 31/40
21/21 [==============================] - 242s 12s/step - loss: 0.0741
Epoch 32/40
21/21 [==============================] - 233s 11s/step - loss: 0.0725
Epoch 33/40
21/21 [==============================] - 232s 11s/step - loss: 0.0710
Epoch 34/40
21/21 [==============================] - 234s 11s/step - loss: 0.0692
Epoch 35/40
21/21 [==============================] - 236s 11s/step - loss: 0.0678
Epoch 36/40
21/21 [==============================] - 236s 11s/step - loss: 0.0662
Epoch 37/40
21/21 [==============================] - 236s 11s/step - loss: 0.0646
Epoch 38/40
21/21 [==============================] - 233s 11s/step - loss: 0.0635
Epoch 39/40
21/21 [==============================] - 235s 11s/step - loss: 0.0628
Epoch 40/40
21/21 [==============================] - 236s 11s/step - loss: 0.0616
```

[54]:
```python
class OneStep(tf.keras.Model):
  def __init__(self, model, chars_from_ids, ids_from_chars, temperature=1.0):
    super().__init__()
    self.temperature = temperature
    self.model = model
    self.chars_from_ids = chars_from_ids
    self.ids_from_chars = ids_from_chars

    # Create a mask to prevent "[UNK]" from being generated.
    skip_ids = self.ids_from_chars(['[UNK]'])[:, None]
    sparse_mask = tf.SparseTensor(
        # Put a -inf at each bad index.
        values=[-float('inf')]*len(skip_ids),
        indices=skip_ids,
        # Match the shape to the vocabulary
        dense_shape=[len(ids_from_chars.get_vocabulary())])
    self.prediction_mask = tf.sparse.to_dense(sparse_mask)

  @tf.function
  def generate_one_step(self, inputs, states=None):
    # Convert strings to token IDs.
    input_chars = tf.strings.unicode_split(inputs, 'UTF-8')
    input_ids = self.ids_from_chars(input_chars).to_tensor()

    # Run the model.
```

```
    # predicted_logits.shape is [batch, char, next_char_logits]
    predicted_logits, states = self.model(inputs=input_ids, states=states,
                                          return_state=True)
    # Only use the last prediction.
    predicted_logits = predicted_logits[:, -1, :]
    predicted_logits = predicted_logits/self.temperature
    # Apply the prediction mask: prevent "[UNK]" from being generated.
    predicted_logits = predicted_logits + self.prediction_mask

    # Sample the output logits to generate token IDs.
    predicted_ids = tf.random.categorical(predicted_logits, num_samples=1)
    predicted_ids = tf.squeeze(predicted_ids, axis=-1)

    # Convert from token ids to characters
    predicted_chars = self.chars_from_ids(predicted_ids)

    # Return the characters and model state.
    return predicted_chars, states

one_step_model = OneStep(model, chars_from_ids, ids_from_chars)
```

[55]:
```
start = time.time()
states = None
next_char = tf.constant(['omg'])
result = [next_char]

for n in range(1000):
  next_char, states = one_step_model.generate_one_step(next_char, states=states)
  result.append(next_char)

result = tf.strings.join(result)
end = time.time()
print(result[0].numpy().decode('utf-8'), '\n\n' + '_'*80)
print('\nRun time:', end - start)
```

```
omg fruedront think he wan she wanted mo to. this is why i am his braid to see
my heade frem because i felt like i fucked up my friendship with duda and
rachael. but now i only even dates this week agazing. i think thats why im just
more of a bomb, and im actually doing writing ppling to me i get better. im
trying harder to learn more about boundaries and trying myself to get hig out
with me. i want to be able to do myself the same but idk if im making yet i mean
so also do it cuz really i only want to and so i definitely shouldnt expect
things from claire ppl. she is just more individualistic than who exactly i am
and geed until that and goga thanks to him kissing him too and they r up all
hella cool as well. eee but… immaknapsime. i also miss my nextss friends
because again.
i just need the koreans so have to fucking god him so many since its more
cerentina and i'm actually being scared to see him again. i have a feeling he
```

```
doesn't like me anymore. i tbl u know i also crave to be able

----------------------------------------------------------------------------

Run time: 4.218449354171753
```

[56]:
```
start = time.time()
states = None
next_char = tf.constant(['omg ', 'omg ', 'omg ', 'omg ', 'omg '])
result = [next_char]

for n in range(100):
  next_char, states = one_step_model.generate_one_step(next_char, states=states)
  result.append(next_char)

result = tf.strings.join(result)
end = time.time()
print(result, '\n\n' + '_'*80)
print('\nRun time:', end - start)
```

```
tf.Tensor(
[b"omg i usud to my onl harging over this awful way. we've been staying dressed
up all night these late pas"
 b"omg in starbucks and they go the lelve our own nsn's… i love watching u
still be more than again his a"
 b'omg i used to just order food off them then mo or i can rlly dont rlly talk
anymore but ig sometimes tha'
 b'omg instead on converstantiny to much how early did i even get over this all
this now but he is angry lo'
 b"omg writing, she jnows what she's good at, or duda is good at coding, and has
the brain for understandin"], shape=(5,), dtype=string)

----------------------------------------------------------------------------

Run time: 1.5143625736236572
```

[57]:
```
# a function to generate more text!
# I just used the code above to put it into a function
def generate_text(model, start_string, num_generate=1000):
    start = time.time()
    states = None
    next_char = tf.constant([start_string])
    result = [next_char]

    for n in range(num_generate):
        next_char, states = model.generate_one_step(next_char, states=states)
        result.append(next_char)
```

```
    result = tf.strings.join(result)
    end = time.time()
    return result[0].numpy().decode('utf-8')

# so I generate 50 new texts
num_texts = 50
generated_texts = [generate_text(one_step_model, start_string='omg ',␣
 ↪num_generate=100) for _ in range(num_texts)]

# and follow the same pre-processing to let my SVM model predict the sentiment
generated_texts_preprocessed = pd.Series(generated_texts).apply(preprocess_text)
generated_texts_tfidf = tfidf_vectorizer.transform(generated_texts_preprocessed)

predictions_svm = model3.predict(generated_texts_tfidf)

# now I attach it to a new dataframe so we can see the results!
df_newtext = pd.DataFrame({'Entry': generated_texts, 'Sentiment':␣
 ↪predictions_svm})
```

[58]:
```
df_newtext.head(15)
```

[58]:

	Entry	Sentiment
0	omg i diend oh okay ytyy daye down to bit but …	Negative
1	omg instead on on of getting me all these ur a…	Negative
2	omg this for cs111 cuz otherwise i just could …	Positive
3	omg so far. i feel like thomas was bad, then w…	Positive
4	omg wrutif shere to stop bazy is alone so but …	Positive
5	omg in some codsetule. i think that vus while …	Positive
6	omg so fare for me interesting but so idk i su…	Positive
7	omg i really just want to stop i wish me becau…	Positive
8	omg insted do me i truly finds peace with with…	Positive
9	omg rls he damases tha way over her will not f…	Positive
10	omg rls. so y but everyone sitting this now it…	Negative
11	omg inso a family, so i hope as you turn 50 th…	Positive
12	omg stuffs. like playbi like this might fit yo…	Positive
13	omg just for undiling my life. i crave stabili…	Negative
14	omg so fare someone so much. it makes me feel …	Positive

5 Overall summary

- First pipeline: Data cleaning and EDA on personal journal entries
  - Tried fitting SVM, Naive Bayes and Logistic regression to myy data, SVM was best in terms of accuracy score and correctly predicting sentiment.
- Second pipeline: Augmenting data and trying to fit data onto new models
  - Data augmentation, conducting PCA to check relationship between entries' sentiments. Trying XGBoost and Random Forest, SVM still best model
- Third pipeline: Building an RNN to then feed into the best model (which I concluded as

SVM) to predict sentiment
  - Building RNN from scratch to write new text in my own voice, to then feed into SVM to predict sentiment of the newly generated RNN text.

When I ran my RNN with more epochs, the text resembled my voice a lot more (at around 30+). But it still didn't generate legible human text that was grammatically correct. Because these texts didn't make sense, I couldn't understand the sentiment scores tagged to the entry text.

To better extend this project, I would need to improve my RNN model by finding more appropriate training data, as it seems like a dataset of 500 is still too small.

6 References

- Hossein Ashtari. (2024, February 21). XGBoost vs. Random Forest vs. Gradient Boosting: Differences | Spiceworks. Spiceworks Inc; Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/xgboost-vs-random-forest-vs-gradient-boosting/#:~:text=XGBoost%2C%20with%20its%20efficient%20regularization,offers%20flexibil
- Text generation with an RNN. (2023). TensorFlow. https://www.tensorflow.org/text/tutorials/text_generation