IMT 547 Project Part II: Data Preprocessing

Chesie Yu

02/18/2024

*This notebook outlines the **data preprocessing** process for the **YouTube Gaming Comment Toxicity** project.*

Components

1. **Data Cleaning**: Data cleaning procedures including handling missing values and converting data types.
2. **Text Preprocessing**: Text cleaning measures including text standardization, irrelevant content removal, stopwords removal, and tokenization.
3. **Data Labeling**: Perspective API toxicity annotations and VADER/TextBlob/Empath sentiment scoring.

Functions

- **`clean(text)`** : Performs text preprocessing steps on a given document.
- **`build_client(api_key)`** : Build a client for a given Perspective API key.
- **`perspective_toxicity(comments)`** : Compute Perspective toxicity scores for a given list of texts. Support throttling management w/ client reuse, key rotation, and exponential backoff.
- **`vader_sentiment(text)`** : Compute VADER sentiment scores for a given text.
- **`textblob_sentiment(text)`** : Compute TextBlob sentiment scores for a given text.
- **`empath_sentiment(text)`** : Compute Empath sentiment scores for a given text.

In [1]:

```python
# Import the libraries
import json
import random
import re
import time

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

import contractions
import nltk
from nltk.corpus import stopwords
import spacy
from spacy_langdetect import LanguageDetector
```

1. Load the Data

In [2]:

```python
# Load the data
yt = pd.read_csv("../data/yt.csv")
yt.head(2)
```

Out[2]:

	channel_id	channel_name	video_id	video_title	video_creation_time	video_description
0	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30T16:40:18Z	🌏 Get exclusive NordVPN deal here ➵ https://N...
1	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30T16:40:18Z	🌏 Get exclusive NordVPN deal here ➵ https://N...

In [3]:

```
# Check the dimensions
print(f"Number of rows: {yt.shape[0]}\n"
      f"Number of columns: {yt.shape[1]}\n")

# Check for missing values
print(f"Number of missing values: {yt.isna().sum().sum()}")
```

```
Number of rows: 140637
Number of columns: 17

Number of missing values: 877
```

The dataset contains ***140,637 comments*** *collected from action and non-action gaming videos on YouTube. It features* ***17 columns*** *on metadata associated with the videos and comments.* ***877 missing entries*** *are detected in this dataset; in the subsequent sections, we will address these data quality concerns.*

2. Data Cleaning

Handle Missing Values

Given that the missing entries account for only ***0.624%*** *of the dataset, we will employ the* ***deletion*** *method to handle these missings. By eliminitating rows that contain missing values, we ensure that our analysis is based on* ***complete and accurate*** *information.*

In [4]:

```
# Check the missings
yt.isna().sum()
```

Out[4]:

```
channel_id                0
channel_name              3
video_id                  3
video_title               3
video_creation_time       3
video_description       789
video_tags                5
video_viewcount           5
video_likecount           5
video_commentcount        5
comment_id                5
comment_author_id         5
comment_text             18
comment_time              7
comment_likecount         7
comment_replycount        7
genre                     7
dtype: int64
```

In [5]:

```
# Remove the missings
yt.dropna(inplace=True)
yt.shape
```

Out[5]:

```
(139833, 17)
```

Convert Data Types

Note that the ***`video_creation_time`*** *and* ***`comment_time`*** *are represented as* ***objects****; since these two columns represent dates and times, we will convert them to the more appropriate type* ***`datetime`*** *for efficient anlaysis.*

In [6]:

```
# Check the data types
yt.dtypes
```

Out[6]:

```
channel_id              object
channel_name            object
video_id                object
video_title             object
video_creation_time     object
video_description       object
video_tags              object
video_viewcount        float64
video_likecount        float64
video_commentcount     float64
comment_id              object
comment_author_id       object
comment_text            object
comment_time            object
comment_likecount      float64
comment_replycount     float64
genre                   object
dtype: object
```

In [7]:

```
# Convert to datetime
yt["video_creation_time"] = pd.to_datetime(yt["video_creation_time"])
yt["comment_time"] = pd.to_datetime(yt["comment_time"])
```

3. Text Preprocessing

Filter English Comments

*To **align** our analysis with the interests of the English-speaking YouTube gaming community, we intend to employ the **spacy-langdetect** tool to **filter our dataset for English comments** only. However, our initial attempt to implement a code solution from SpaCy's documentation was unsuccessful; if time permits, we will explore alternative methods to isolate English comments for our analysis.*

In [8]:

```
# # Load the SpaCy model
# # Documentation: https://pypi.org/project/spacy-langdetect/
# nlp = spacy.load("en_core_web_sm")
# nlp.add_pipe(LanguageDetector(), name="language_detector", last=True)

# def filter_english(comment):
#     """
#     Detect English comments.
#     """
#     doc = nlp(comment)
#     return doc._.languege["language"] == "en" and doc._.language["score"] > 0.95

# yt = yt[yt["comment_text"].apply(filter_english)]
# yt.shape
```

Text Cleaning

*To preserve the **most relevant information**, we will undertake a series of text preprocessing steps to refine our corpus for analysis.*

*This initial step involves **text standardization** to ensure that the text will be **consistently understood** by analytical tools. All texts will be converted to **lowercase**; **contractions** will be expanded to their full forms using the `contractions` library.*

*Next, we will **remove the URLs, mentions, hashtags, and non-alphabetic characters** to eliminate the noise in data. Common English **stopwords** will also be removed as they do not possess significant information. Note the **potential caveat** in this procedure: the elimination of these elements could result in loss of certain nuances in text.*

In [9]:

```
# Function for text preprocessing
def clean(text):
    """
    Performs text preprocessing steps on a given document.
    """
    # Convert to lowercase
    text = text.lower()
    # Remove contractions
    text = contractions.fix(text)

    # Remove URLs
    text = re.sub(r"http\S+", "", text)
    # Remove mentions
    text = re.sub(r"(?<![@\w])@(\w{1,25})", "", text)
    # Remove hashtags
    text = re.sub(r"(?<![#\w])#(\w{1,25})", "", text)
    # Remove new line characters
    text = re.sub("\n", " ", text)

    # Remove non-alphabetic characters
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    # Remove extra spaces
    text = re.sub(r"\s+", " ", text)

    # Remove stop words
    stop_words = set(stopwords.words("english"))
    text = " ".join([word for word in text.split() if word not in stop_words])

    return text
```

In [10]:

```
# Extract the comments
comments = yt["comment_text"]
comments[:5]
```

Out[10]:

```
0    Damn dude, even with mimic I think it would ta...
1    This is the pewds that I thought he’d turn int...
2    This is actually awesome. Can't believe a meme...
3    Wow, didn't even know Pewds had this analytica...
4    Damn, i cant believe it took me 11 months afte...
Name: comment_text, dtype: object
```

In [11]:

```
# Clean the comments
comments = comments.apply(clean)

# Remove empty comments
comments = comments[comments.str.len() > 0]
```

Tokenization

*Using `word_tokenizer`, we will **tokenize** the text into smaller pieces. This process will be crucial for **analyzing term frequency** or **identifying common themes** within the corpous as the analysis progresses.*

In [12]:

```
# Import the libraries
from nltk.tokenize import word_tokenize

# Tokenize the comments
tokenized_comments = comments.apply(word_tokenize)
```

In [13]:

```
# Combine into one DataFrame
yt["cleaned_comment"] = comments
yt["tokenized_comment"] = tokenized_comments
yt.head()
```

Out[13]:

	channel_id	channel_name	video_id	video_title	video_creation_time	video_description
0	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
1	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
2	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
3	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
4	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...

In [14]:

```
# Remove the missings
yt.dropna(inplace=True)
yt.shape
```

Out[14]:

```
(138996, 19)
```

In [15]:

```
# Write to CSV
yt.to_csv("../data/yt_cleaned.csv", index=False)
```

4. Data Labeling

Toxicity Annotations

Acquiring the toxicity labels is crucial for analyzing toxicity in comments. However, manually annotating nearly 140,000 comments is ***impractical*** *given the large volume and resource limitations. Thus, to effectively* ***quantify the level of toxicity*** *in comments, we will leverage the* ***Perspective API*** *to obtain our true labels.*

Quota Limits and Throttling Management

The Perspective API, however, enforces a ***quota limit*** *of* ***1 query per second (QPS)*** *for each project. Despite the* ***lack of batch processing*** *support, we have devised a* ***throttling management*** *strategy that incorporates* ***key rotation*** *and* ***exponential backoff*** *to efficiently manage this constraint.*

Our approach involves cycling through ***10 different API keys*** *and their respective* ***pre-built clients****, enhancing our query capacity within the API's quota restrictions. Furthermore, an* ***exponential backoff*** *mechanism is enforced to manage* ***retries*** *following any quota breaches or server errors. This method* *will **systematically increase the delay between subsequent requests**, thereby minimizing the likelihood of succesive failures and mitigating the impact on the API server.*

*Additional features such as **logging** and **exception handling** are integrated to support **monitoring** and **troubleshooting**, facilitating a smooth and efficient data lebeling process. These measures collectively **reduce the projected processing time** from an initial estimate of **2.26 days** to approximately **4 hours**.*

In [16]:
```
# Import the libraries
import itertools
import logging
from googleapiclient import discovery
from googleapiclient.errors import HttpError
```

In [17]:
```
# The Perspective API keys
PERSPECTIVE_API_KEYS = [
    "AIzaSyAMpL8JpwPU4c1nEGKCiBAiGp979r6o4-4",  # perspective-api-414709
    "AIzaSyD_-Oiitvk4OL5zgvX90Nn5TcoA23TrMlM",  # perspective-api-414723
    "AIzaSyCLQ0SAdw0-xKDEqGyTcBPO7yApPF2M3R0",  # perspe-414800
    "AIzaSyDTzo_CBwQ_5zVDojWSBMnH1jI_F6rEs7s",  # precise-antenna-414801
    "AIzaSyAt70Atcrnx2bfvFuPTwtvOV8Nf2PBPx4A",  # sound-datum-414801
    "AIzaSyBgO09nuuysiO7YNqexVZiskWhJPSv5t3A",  # perspective-api-414710
    "AIzaSyBFU4rFCLaCAVuQ0i4K3QhF_f9wBV4gBm4",  # perspective-api-414800
    "AIzaSyC8kMo6iX7iXX_lj8gx8IM0LuNS8p94UA4",  # shaped-canyon-414800
    "AIzaSyAhRHCYoYkRkQkco4NzhNuKT7Zm92BKOS8",  # perspective-api-414801
    "AIzaSyCr_b9CLWmy9Rt0f0ME74ZZmh3uT6gAwpk"  # hardy-order-414801
]

def build_client(api_key):
    """
    Build a client for a given Perspective API key.
    """
    # Create a client object
    # Reference: https://developers.google.com/codelabs/setup-perspective-api#4
    client = discovery.build(
        "commentanalyzer",  # Name
        "v1alpha1",  # Version
        developerKey=api_key,
        discoveryServiceUrl="https://commentanalyzer.googleapis.com/$discovery/rest?versi
        static_discovery=False
    )
    return client

# Pre-build a client for each API key
clients = {key: build_client(key) for key in PERSPECTIVE_API_KEYS}

# Set up the iterator
api_key_iterator = itertools.cycle(PERSPECTIVE_API_KEYS)
```

In [18]:
```
# Configure logging to file
logging.basicConfig(
    filename="../logs/toxicity.log",
    level=logging.INFO,  # Log info, warning, error, critical
    format="%(asctime)s - %(levelname)s - %(message)s",
    filemode="w"  # Overwrite on each run
)
```

In [19]:
```
def perspective_toxicity(comments):
    """
    Compute Perspective toxicity scores for a given list of texts.
    Support throttling management w/ client reuse, key rotation, and
    exponential backoff.
    """
    # Empty list to store toxicity scores
    scores = []

    # Loop through the comments
    for index, comment in enumerate(comments):
        # Specify the comment text and attributes
        analyze_request = {
            "comment": {"text": comment},
            "languages": ["en"],
            "requestedAttributes": {
                "TOXICITY": {},
                "SEVERE_TOXICITY": {},
                "IDENTITY_ATTACK": {},
                "INSULT": {},
                "PROFANITY": {},
                "THREAT": {}}
        }

        # Attempts allowed
        attempts_per_key = 5
        total_attempts = len(PERSPECTIVE_API_KEYS) * attempts_per_key
        # Reset attempt count for each comment
        attempt = 0

        # While retry attempts are not exhausted
        while attempt < total_attempts:
            # Rotate to the next API key
            current_key = next(api_key_iterator)
            client = clients[current_key]

            try:
                res = client.comments().analyze(body=analyze_request).execute()
                scores.append({
                    "toxicity": res["attributeScores"]["TOXICITY"]["summaryScore"]["value
                    "severe_toxicity": res["attributeScores"]["SEVERE_TOXICITY"]["summary
                    "identity_attack": res["attributeScores"]["IDENTITY_ATTACK"]["summary
                    "insult": res["attributeScores"]["INSULT"]["summaryScore"]["value"],
                    "profanity": res["attributeScores"]["PROFANITY"]["summaryScore"]["val
                    "threat": res["attributeScores"]["THREAT"]["summaryScore"]["value"]
                })
                logging.info(f"Success for comment #{index} with key {current_key} on att
                # Break the loop if successful
                break

            # Http errors
            except HttpError as e:
                # Rate limit errors
                if e.resp.status == 429:
                    logging.warning(f"HTTP 429 Rate limit exceeded for comment #{index} w
                else:
                    logging.warning(f"HTTP error for comment #{index} with key '{current_
            # Timeout errors
            except TimeoutError:
                logging.warning(f"TimeoutError for comment #{index} with key '{current_ke
            # Unexpected errors
            except Exception as e:
                logging.warning(f"Unexpected error for comment #{index} with key '{curren

            # Exponential backoff + random jitter
            sleep_time = (2 ** (attempt // len(PERSPECTIVE_API_KEYS))) + random.uniform(0
            time.sleep(sleep_time)
            attempt += 1

            # Check if all retry attempts are exhausted
            if attempt >= total_attempts:
                logging.error(f"Max attempts reached for comment #{index} with key {curre

        # Sleep to avoid exceeding rate limits
        # time.sleep(0.05)

    # Convert to DataFrame
    toxicity_scores = pd.DataFrame(scores)

    return toxicity_scores
```

In [20]:

```
# %%timeit -r 1 -n 3
# Start timing
start_time = time.time()

# Compute Perspective API toxicity scores for each comment
toxicity_scores = perspective_toxicity(comments)

# End timing
print(f"Runtime: {time.time() - start_time:.4f}")
toxicity_scores.head()
```

```
Runtime: 14948.2070
```

Out[20]:

	toxicity	severe_toxicity	identity_attack	insult	profanity	threat
0	0.642621	0.169603	0.044097	0.342037	0.600193	0.138155
1	0.093515	0.004025	0.012943	0.023351	0.038906	0.009515
2	0.201028	0.011749	0.016059	0.025929	0.098687	0.106963
3	0.137353	0.007057	0.013345	0.028061	0.060951	0.013217
4	0.509388	0.120196	0.034301	0.249039	0.498944	0.014566

In [21]:

```
# Combine into one DataFrame
for column in toxicity_scores.columns:
    yt[column] = toxicity_scores[column].values
yt.head()
```

Out[21]:

	channel_id	channel_name	video_id	video_title	video_creation_time	video_description
0	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
1	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
2	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
3	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
4	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...

5 rows × 25 columns

In [22]:

```
# Check the dimensions
yt.shape
```

Out[22]:

```
(138996, 25)
```

In [23]:

```
# import requests
# import json

# # The URL for the Perspective API
# url = "https://commentanalyzer.googleapis.com/v1alpha1/comments:analyze?key=" + PERSPEC

# # The data sent to request
# data_dict = {
#     "comment": {"text": "Friendly discussion is cool, but please no personal attacks!"}
#     "languages": ["en"],
#     "requestedAttributes": {"TOXICITY": {}}
# }

# response = requests.post(url, data=json.dumps(data_dict))
# result = response.json()

# print(result)
```

Sentiment Scoring

*To further investigate the **emotional dynamics** of the comments, we will generate the **sentiment scores** using **VADER**, **TextBlob**, and **Empath**. Note that our initial analysis with Empath will concentrate on positive and negative emotions; yet if time allows, we hope to extend our examination to encompass* *all Empath categories in the future, aiming for a more nuanced understanding of the prevalent themes within YouTube gaming comments.*

In [24]:

```
# Import the libraries
from nltk.corpus import opinion_lexicon
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
from empath import Empath
```

In [25]:

```
def vader_sentiment(text):
    """
    Compute VADER sentiment scores for a given text.
    """
    # Initialize the analyzer
    analyzer = SentimentIntensityAnalyzer()

    # Compute the scores
    return analyzer.polarity_scores(text)
```

In [26]:

```
# %%timeit -r 1 -n 1
# Compute VADER sentiment scores for each comment
vader_scores = comments.apply(vader_sentiment).apply(pd.Series)
vader_scores.head()
```

Out[26]:

	neg	neu	pos	compound
0	0.315	0.572	0.113	-0.6395
1	0.000	0.703	0.297	0.5859
2	0.124	0.442	0.434	0.7906
3	0.000	0.549	0.451	0.9324
4	0.093	0.687	0.220	0.5709

In [27]:

```
def textblob_sentiment(text):
    """
    Compute TextBlob sentiment scores for a given text.
    """
    # Initialize the analyzer
    blob = TextBlob(text)

    # Compute the scores
    return {"polarity": blob.sentiment.subjectivity,
            "subjectivity": blob.sentiment.subjectivity}
```

In [28]:

```
# Compute TextBlob sentiment scores for each comment
textblob_scores = comments.apply(textblob_sentiment).apply(pd.Series)
textblob_scores.head()
```

Out[28]:

	polarity	subjectivity
0	0.400000	0.400000
1	0.345238	0.345238
2	0.583333	0.583333
3	0.560000	0.560000
4	0.675000	0.675000

In [29]:

```
def empath_sentiment(text):
    """
    Compute Empath sentiment scores for a given text.
    """
    # Initialize the analyzer
    lexicon = Empath()

    # Compute the scores
    categories = lexicon.analyze(text, normalize=True)

    # Filter out the positive and negative emotions
    return {k:v for k, v in categories.items() if k in ["positive_emotion", "negative_emo
```

In [30]:

```
# Compute Empath sentiment scores for each comment
empath_scores = comments.apply(empath_sentiment).apply(pd.Series)
empath_scores.head()
```

Out[30]:

	negative_emotion	positive_emotion
0	0.066667	0.000000
1	0.000000	0.100000
2	0.071429	0.000000
3	0.000000	0.000000
4	0.000000	0.043478

In [31]:

```
# Combine into one DataFrame
yt = pd.concat([yt, vader_scores, textblob_scores, empath_scores], axis=1)
yt.head()
```

Out[31]:

	channel_id	channel_name	video_id	video_title	video_creation_time	video_description
0	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
1	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
2	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
3	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...
4	UC-lHJZR3Gqxm24_Vd_AJ5Yw	PewDiePie	F-yEoHL7MYY	I ~~tried to~~ beat Elden Ring Without Dyi...	2022-04-30 16:40:18+00:00	🌏 Get exclusive NordVPN deal here ➼ https://N...

5 rows × 33 columns

In [32]:

```
# Check the dimensions
yt.shape
```

Out[32]:

```
(138996, 33)
```

In [33]:

```
# Write to CSV
yt.to_csv("../data/yt_labeled.csv", index=False)
```

*The labeled dataset contains **138,996 rows** and **33 columns**. In `03-preliminary anlaysis`, we will begin to explore the dataset, examining its **distribution** through **exploratory data analysis** and **visualizations**.*