Gaming and Political Extremism

Virginia Tech, Blacksburg, VA 24061

Instructor: Dr. Edward A. Fox

Client: Dr. James Ivory, Michael Senters

Team: Niko Olson, Mauricio Lovelace, Earl Carter
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Figures</td>
<td>2</td>
</tr>
<tr>
<td>Table of Tables</td>
<td>3</td>
</tr>
<tr>
<td>1.0 Abstract</td>
<td>4</td>
</tr>
<tr>
<td>2.0 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Client</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Problem</td>
<td>5</td>
</tr>
<tr>
<td>2.3 General Approach</td>
<td>5</td>
</tr>
<tr>
<td>3.0 Requirements</td>
<td>6</td>
</tr>
<tr>
<td>4.0 Design</td>
<td>7</td>
</tr>
<tr>
<td>5.0 Implementation</td>
<td>7</td>
</tr>
<tr>
<td>5.1 Frontend</td>
<td>7</td>
</tr>
<tr>
<td>5.2 Backend</td>
<td>8</td>
</tr>
<tr>
<td>6.0 Data Collection, Processing, and Findings</td>
<td>9</td>
</tr>
<tr>
<td>6.1 Data Sources</td>
<td>9</td>
</tr>
<tr>
<td>6.2 Data Collection Methodology</td>
<td>9</td>
</tr>
<tr>
<td>6.3 Data Structure</td>
<td>11</td>
</tr>
<tr>
<td>6.4 Data Cleaning and Preprocessing</td>
<td>11</td>
</tr>
<tr>
<td>6.5 Data Visualization</td>
<td>12</td>
</tr>
<tr>
<td>6.6 Data Findings and Graphs</td>
<td>14</td>
</tr>
<tr>
<td>6.7 Analysis of Findings</td>
<td>17</td>
</tr>
<tr>
<td>7.0 Evaluation</td>
<td>18</td>
</tr>
<tr>
<td>7.1 Evaluation of Data Collection and Processing</td>
<td>18</td>
</tr>
<tr>
<td>7.2 Findings and Limitations</td>
<td>18</td>
</tr>
<tr>
<td>8.0 User’s Manual</td>
<td>18</td>
</tr>
<tr>
<td>8.1 Supported Functionality</td>
<td>18</td>
</tr>
<tr>
<td>9.0 Developer’s Manual</td>
<td>23</td>
</tr>
<tr>
<td>9.1 Inventory of all figures</td>
<td>23</td>
</tr>
<tr>
<td>9.2 Inventory of all relevant files</td>
<td>24</td>
</tr>
<tr>
<td>10.0 Lessons Learned</td>
<td>24</td>
</tr>
<tr>
<td>10.1 Timeline/Schedule</td>
<td>25</td>
</tr>
<tr>
<td>10.2 Problems</td>
<td>26</td>
</tr>
<tr>
<td>10.3 Solutions</td>
<td>27</td>
</tr>
<tr>
<td>10.4 Future Work</td>
<td>27</td>
</tr>
<tr>
<td>11.0 Acknowledgements</td>
<td>28</td>
</tr>
<tr>
<td>12.0 References</td>
<td>28</td>
</tr>
<tr>
<td>13.0 Appendices</td>
<td>30</td>
</tr>
<tr>
<td>13.1 Appendix A: Methodology Assignment</td>
<td>30</td>
</tr>
</tbody>
</table>
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total Views on Arma 3 YouTube Videos</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Figure 2: Line Graph of Total Likes on Arma 3 YouTube Videos</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Sample Data Collected on Google Sheets</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Screenshot of Scraper code</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Screenshot of Cleaner.py</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Screenshot of visualizer.py</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Line graph of YouTube Views over Time of MilSim Games</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Bar Chart of Arma 2 Views over time</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>Bar Chart of Arma 3 Views over time</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>Bar Chart of Gamergate Views over time</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>Bar Chart of Rising Storm 2 Vietnam Views over time</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>Bar Chart of War of Rights Views over time</td>
<td>17</td>
</tr>
<tr>
<td>13</td>
<td>Accessing Google App Script</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>Aspects that Client can change outlined in Red</td>
<td>20</td>
</tr>
<tr>
<td>15</td>
<td>Sample Execution Log output</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>Sample Data after Executing Data Scraper</td>
<td>22</td>
</tr>
<tr>
<td>17</td>
<td>Developing Environment for scrape</td>
<td>23</td>
</tr>
<tr>
<td>18</td>
<td>Collection Data Diagram</td>
<td>33</td>
</tr>
<tr>
<td>19</td>
<td>Analyzing Data Diagram</td>
<td>33</td>
</tr>
<tr>
<td>20</td>
<td>Visualizing Data Diagram</td>
<td>34</td>
</tr>
</tbody>
</table>
Table of Tables

Table 1: Script, repository, and visualization files.............................................................24
Table 2: Timeline/Schedule...............................................................................................25
Table 3: Services Table......................................................................................................35
1.0 Abstract

The Gaming and Political Extremism project is an in-depth research initiative that investigates the dynamics of online group interactions within military simulation (MilSim) gaming communities. The project, which is IRB/HRRP-approved, focuses on political and ideological extremism content in MilSim games.

To understand the dynamics of online group interactions within military simulation gaming communities and to specifically find whether engagement with military simulation game content has increased after GamerGate and the 2016 Trump election, we collected YouTube data over the past 10 years from 2013 to 2023. To do so, we utilized Google Sheets and the YouTube API to develop a data-scraping tool targeting the popular MilSim games, with our focus on Arma 2, Arma 3, Rising Storm 2: Vietnam, and War of Rights.

Our team successfully collected data on these MilSim gaming videos, but this was not without its problems. We faced several challenges during the data collection process, including the limitation of only scraping fifty entries per page of results, and issues with collecting relevant data in terms of region and language. To overcome these challenges, we developed solutions such as creating a for loop to collect data from multiple pages and filtering videos based on language and region of origin.

However, this proved to be more challenging than what we initially thought, and then we began cleaning the data. Through this process, we refined our web scraping skills, improved our data collection methods, and addressed challenges related to data filtering and language barriers as mentioned. Once we finished, we obtained a curated dataset for empirical analyses, a summary of findings suitable for sharing with a public audience, and the code for collecting the dataset.

For the final remaining work, our team focused on deciding on and creating visualizations, and finalizing the project report and presentations. The cleaned dataset and visualizations will be used to present our findings in a manner that is easily understood by the general public.

Throughout the project, our team learned valuable lessons, such as efficient time management, communicating with clients, and self-learning new concepts, such as web scraping and new data visualization tools. We also gained expertise in working with the YouTube API and data-scraping tools. This project has not only provided our team with valuable experience but also paved the way for future research on the topic, potentially leading to published findings and an accessible public narrative.
2.0 Introduction

Online military simulation (MilSim) gaming groups have been a popular form of entertainment for many individuals, with games such as Arma 2 [1], Arma 3 [2], Rising Storm 2: Vietnam [3], and War of Rights [4] gaining significant followings. MilSim games allow players to simulate military operations and engage in team-based gameplay. However, recent events have raised concerns about the potential links between online gaming groups and political extremism [5].

This project aims to systematically examine the dynamics of online group interactions in communities centered around MilSim games, with a particular focus on political and ideological content in discussions, including content related to political extremism. The project seeks to determine whether engagement with MilSim game content has increased after events such as Gamergate [6] and the 2016 Trump election.

2.1 Client

The client for this project is the School of Communications at Virginia Tech. The project is being led by Dr. James D. Ivory, a professor in the department and head of the VT G.A.M.E.R. Lab [7]. More specifically, we are aiding a graduate student named Michael Senters in collecting and visualizing data for his thesis.

2.2 Problem

This project aims to address the problem of potential links between online military simulation (MilSim) gaming communities and political extremism by investigating the dynamics of group interactions within these communities.

2.3 General Approach

To address this problem, the project will use data-scraping tools to collect data on online group interactions in MilSim gaming communities, with a particular focus on political and ideological content. The data will be collected using the YouTube API and the Google Script App, and will be curated for empirical analysis. The project will also have a heavy focus on games such as Arma 2 and 3, Rising Storm: Vietnam, and War of Rights.
The project will be led by Dr. Ivory and will involve collaboration with graduate students and other collaborators to determine the sampling frame for data collection and to prepare the data for analysis. The project will produce a data set for empirical analyses, as well as a summary of findings suitable for sharing with a public audience in the form of a PowerPoint, paper, or website. The project will also deliver the code for collecting the dataset.

3.0 Requirements

List of data requirements from YouTube Milsim Videos:

- Ideally collect MilSim video data from the past 10 years (2013 - 2023)
  - Especially data from 2 years prior to Gamergate (2015) and the Trump 2016 Election
- If time frame is too large, start scraping from 2013 and work towards the present

MilSim Game Titles to collect data from:

- Arma 2
- Arma 3
- Rising Storm 2: Vietnam
- War of Rights

Data to collect from the MilSim Videos:

- Video Title
- Number of likes
- Number of comments
- Creator Name
- Number of views – minimum views if necessary
- Tags: age restrictions, demonetized, private
- Region restrictions for primarily English speaking countries
  - Includes: US, Canada, Australia, New Zealand, UK
- Language Restrictions: to English if necessary
4.0 Design

For the design of our solution to our client’s problems, we decided to create a web scraper that would utilize some form of YouTube API, as suggested by our client. Upon further inspection, our team was able to form a rough idea of how we wanted to structure our first draft of code. In order to acquire the results we were looking for, we needed to acquire data related to the videos such as the number of likes, comments, views, and publication date. The first draft of code was able to gather this data successfully, but the code only produced fifty results at a time. In order to become more efficient, we decided to create another draft of code that would produce a great deal more. Our team also decided with the client that the results should not include videos from other countries or languages, so this was another requirement we had to consider. After more research on the YouTube API, we found that this was accomplishable through a filtering specification. This filtering aspect played an integral role in our design, as it was able to effectively remove the irrelevant videos that may have titles matching the search result due to discrepancies between languages. Our team also decided on creating the script in a way that would collect the results by page, with each page of YouTube results containing fifty results. This was accomplished through the use of a for loop that would execute until the number of pages specified was reached. After we had reached this point, the design section of our process had come to an end.

5.0 Implementation

The implementation of this project is divided into two main components: frontend and backend. The project follows a timeframe, with data scraping and data cleaning (backend) planned to finish by mid to late March and transitioning to creating visualizations (frontend) in April.

5.1 Frontend

The frontend consists of data visualizations, which will provide the necessary graphical representations to support the client’s thesis. Using the Pandas [8] API, histograms and line graphs will be created to depict engagement with MilSim games in relation to major events. Graphs showcasing average views, comments, and likes over time will help demonstrate these relationships. Comparison charts across multiple games will also aid in understanding the data collected. If the client decides to publish their findings on a website, assistance will be provided in creating the frontend for the site.

Figure 1 and Figure 2 shows example visualization using sample data from Arma 3:
5.2 Backend

The backend involves a web-scraping script to collect data from YouTube. Google provides an API [10] and scripting language [11] to facilitate this process. The API offers various tools and features for fine-tuning data scraping, allowing the extraction of specific data on the three MilSim games of interest from 2013 to 2023 within English-speaking countries. Parameters like 'country_code', 'start_date', 'end_date', etc., help narrow the search. All collected data is automatically written to a Google Sheets file in the format shown in Figure 3.
These Google Sheets can then be downloaded as CSV files which can then be processed through our Python cleaning script. The Python cleaning script removes duplicate videos by removing any entries with ‘Video ID’s that appear earlier in the file. The cleaning script also removes any videos with ‘Publish Dates’ from before 2013, which was a time range that was requested by our client.

6.0 Data Collection, Processing, and Findings

6.1 Data Sources

In this project, we primarily collected data from YouTube, focusing on videos related to the MilSim games Arma 2 & 3, Rising Storm 2: Vietnam, and War of Rights. These sources were chosen because they provide a comprehensive view of the engagement and dynamics within the MilSim gaming community.

6.2 Data Collection Methodology

We used a web-scraping script to collect data from YouTube, leveraging Google's API and scripting language to streamline the process. The API offers a range of tools and features, allowing us to fine-tune our scraping process. To meet the project’s objectives, we focused on videos about the three MilSim games published between 2013 and 2023 within English-speaking
countries. We set parameters such as 'country_code', 'start_date', and 'end_date' to refine our search accordingly.

To provide a better understanding of our data collection process, we share the Google Apps Script code snippet used to scrape YouTube video data as shown on Figure 4, which performs the scraping based on a search query and writes the data to a Google Sheet.

```
function YouTubeScraper(iterations, search_query, sheet_index) {
  var spreadsheet = SpreadsheetApp.getActiveSpreadsheet()
  var activeSheet = spreadsheet.getActiveSheet()
  if (sheet_index == 0) {
    activeSheet = spreadsheet.getActiveSheet()
  }
  var next = null
  var start_date = new Date('2013-01-01 00:00:00').toString();
  var end_date = new Date().toString();
  for (let i = 0; i < iterations; i++) {
    if (next == null) {
      var search = YouTube.Search.list('snippet', id, { q: search_query, maxResults: 50, relevanceLanguage: 'en', regionCode: 'US', startDate: start_date, endDate: end_date });
    } else {
      var search = YouTube.Search.list('snippet', id, { q: search_query, maxResults: 50, pageToken: next, relevanceLanguage: 'en', regionCode: 'US', startDate: start_date, endDate: end_date });
    }
    var results = search.items.map(item => [item.id, item.snippet.title, item.snippet.publishedAt])
    next = search.nextPageToken
    var ids = results.map(id => id[0]).join(','),
    var stats = YouTube.Videos.list('statistics', { id: ids })
    var videoStats = stats.items.map(item => [item.id, item.statistics.viewCount, item.statistics.likeCount, item.statistics.commentCount])
    activeSheet.getRange(2 + [50 * i + 1], 1, results.length, results[0].length).setValues(results)
    activeSheet.getRange(2 + [50 * i + 1], 4, videoStats.length, videoStats[0].length).setValues(videoStats)
    // console.log(videoStats.length)
    // console.log(videoStats[0].length)
  }
  return formatDateString(date, Session, 'yyyy-MM-dd');
}
```

Figure 4: Screenshot of Scraper code

This code in Figure 4 is a Google Apps Script in JavaScript that scrapes YouTube video data based on a search query and writes the data to a Google Sheet:

- Defines a YouTubeScraper function that takes three arguments: iterations, search_query, and sheet_index.
- Gets the active Google Sheet and sets the active sheet based on the sheet_index parameter. If sheet_index is 0, the script uses the currently active sheet.
- Defines the start_date and end_date variables to search for videos published between January 1, 2013 and the current date.
• Executes a loop based on the iterations parameter to search for YouTube videos using the provided search_query. The script uses the YouTube Data API to retrieve search results in batches of 50.
• If there is a nextPageToken, the script performs another search using the token to get the next batch of results.
• Extracts the video IDs, titles, and published dates from the search results.
• Retrieves video statistics, including view count, like count, and comment count, for each video ID.
• Writes the video data and statistics to the active sheet in the Google Sheet.
• Defines a formatDateString function that takes a date and returns a formatted date string.
• Defines an array of search queries (queries) and loops through each query, calling the YouTubeScraper function and the current index as the sheet_index.

6.3 Data Structure

The collected data included variables such as video titles, like counts, view counts, number of comments, creator names. The data was stored in Google Sheets in a structured format, with each row representing a video and each column containing a specific variable.

6.4 Data Cleaning and Preprocessing

After collecting the data using the web scraping script, the next step is to clean and preprocess the data to ensure its quality and suitability for analysis. The following Python code in Figure 5 uses the Pandas library to perform data cleaning.

```python
import pandas as pd
import csv as csv
import datetime as dt
import numpy

file_name = input("Input File Name:\t")
file_name_output = input("Output File Name:\t")

df = pd.read_csv(file_name, sep="", engine='python')
df["Publish Date"] = pd.to_datetime(df["Publish Date"])

df = df[df["Publish Date"].dt.year >= 2013]
df = df[df["Publish Date"].dt.year >= 2013]
cleanedCsv = df.drop_duplicates(["Video ID"])
cleanedCsv.to_csv(file_name_output, index=False)
```

Figure 5: Screenshot of Cleaner.py

This code in Figure 5 performs the following tasks:
1. Imports the necessary libraries, such as Pandas, CSV, and datetime.

2. Reads the input CSV file containing the raw data.

3. Converts the 'Publish Date' column to a datetime format.

4. Filters the data to include only videos published between 2013 and the present.

5. Removes duplicate video entries based on the 'Video ID' column.

6. Writes the cleaned data to a new CSV file.

By cleaning and preprocessing the data, we ensure that the analysis and visualizations are based on accurate and relevant information.

6.5 Data Visualization

Once the data is cleaned and preprocessed, it is essential to create visualizations to effectively communicate the findings of the project. The following Python code in Figure 6 uses the Plotly and Pandas libraries to generate interactive visualizations.
This code in Figure 6 performs the following tasks:

1. Imports the necessary libraries, such as Pandas, Plotly, and Cufflinks.

2. Defines the input file names and game names.

3. Initializes an empty Plotly Figure.

4. Creates a list of DataFrames by reading each cleaned CSV file.
5. Sorts the data by 'Publish Date' in ascending order.

6. Generates a line chart with markers showing the YouTube views over time for each game.

7. Updates the layout and formatting of the line chart.

8. Displays the line chart.

9. Generates and displays a histogram for each game, showing the average views over time.

These visualizations help identify trends and patterns or lack thereof in the data, enabling the project stakeholders to gain insights into the dynamics of online group interactions in MilSim gaming communities and potential links to political extremism.

6.6 Data Findings and Graphs

Using Pandas and Plotly on our collected and processed data, we were able to create different visualizations and graphs to better understand the data. The graphs we decided to include are shown in Figure 7-12:

![Figure 7: Line graph of YouTube Views over Time of MilSim Games](image)

Figure 7: Line graph of YouTube Views over Time of MilSim Games
Figure 8: Bar Chart of Arma 2 Views over time

Figure 9: Bar Chart of Arma 3 Views over time
Figure 10: Bar Chart of Gamergate Views over time

Figure 11: Bar Chart of Rising Storm 2 Vietnam Views over time
Figure 12: Bar Chart of War of Rights Views over time

6.7 Analysis of Findings

Based on the data and graphs generated, including Figure 7-12, our initial thesis is supported to some extent. The question was whether engagement with MilSim game content increased after events like Gamergate and the Trump election. Although we observed an increase in engagement, it occurred well after the peak of these events, making it unclear whether it’s a correlation or causation.

Gamergate began in August 2014, with the most intense period spanning from late 2014 to early 2015 [12]. The Trump election took place between his candidacy announcement on June 16, 2015, [13] and his inauguration as the 45th US President on January 20, 2017. Figure 7 demonstrates increased engagement in MilSim gaming communities for all games after these events. However, we cannot definitively conclude if there is a cause-and-effect relationship or if it’s merely a correlation between the events.
7.0 Evaluation

7.1 Evaluation of Data Collection and Processing

The data collection and processing methods used in this project proved to be successful in gathering a substantial amount of relevant YouTube video data related to MilSim games. The Google Apps Script enabled the retrieval of video information, including video IDs, titles, published dates, view counts, like counts, and comment counts, filtered based on specified search queries and date ranges. The Python scripts for cleaning and analyzing the data were also effective in removing duplicate entries, filtering out irrelevant videos, and formatting the data for visualization purposes.

7.2 Findings and Limitations

Our findings indicate a possible link between the Gamergate and Trump election events and increased engagement in MilSim gaming communities. However, the increase in engagement appears to be delayed, occurring well after the peak of these events, making it unclear whether it’s a correlation or causation.

While our methods were successful in collecting and analyzing a large volume of data, there are limitations to consider. The analysis only focused on YouTube video data and may not capture the full extent of engagement across other platforms or social media channels. Additionally, the data collected might not be exhaustive, as the API search results were limited to 100 results per topic.

8.0 User’s Manual

8.1 Supported Functionality

The web scraper our team utilized is embedded within an extension of Google Sheets. This extension allows us to create scripts and scrape data as needed. To access it, first set up a Google Sheet to store the scraped data. Open Google Apps Script by navigating to "Extensions" > "Apps Script" in the Google Sheets menu.
After accessing the script through the Apps Script extension, copy the provided code snippet from the "Data Collection Methodology" section into the script editor. The user is then able to specify the title of the video they are looking for, the language of the videos they are looking for, the region of origin, and the number of results they wish to acquire.
In Figure 14, the areas the client can change in order to specify the results they are looking for are underlined in red. The first underlined section in the search is for the title, the next section is for the relevant language, and the next underlined section specifies the region of origin. The underlined parameter for the YouTubeScraper call is the iterations variable. The iterations variable is a user-specified number that tells the scraper how many results it needs to collect. We can see in Figure 14 that the last line of code is “YouTubeScraper(20)”. This means that the user has specified an iteration value of twenty. Each iteration of scraping produces fifty results, meaning the user would like to find a total of one-thousand results for the game they specified, as twenty iterations of fifty is equal to one thousand results.

After the user changes the script in order to acquire the results they are looking for, they are then able to hit the run button. Once the run button is hit, the script will run through the iterations, scrape the relevant data, and save it into the spreadsheet from which the extension was launched.
Figure 15 shows what the execution of the script will look like from the Google Apps Script extension point of view. The run button that the user uses to execute the script can be found in the box in red, and the box in blue displays the execution log. The execution log will let the reader know when the script starts and finishes running and will also display any errors such as syntax errors or runtime errors with the execution, should they occur. For more information on how we troubleshoot App Scripts we used its Troubleshooting page [9].

Once the script has been executed, the user is then able to navigate back to the Google Sheet they started on, which will then be filled with the data they requested from the script. At this point the user is able to manipulate the data as they see fit in order to produce visualizations or conduct a data analysis.
Figure 16: Sample Data after Executing Data Scraper

Figure 16 displays what a Google Sheet will look like after a single execution of the data scraping script. Each column contains data related to the videos, such as the title, publish date, view count, like count, and comment count. At this point, the user can run the script once more for each of the games they are interested in acquiring data for. Seeing as how many mil-sim games have very specific titles, we can reliably count on the scraper to return relevant data when it returns video results that contain the query specified by the user in each of the videos' titles. In order to find every possible result, the user would have to specify a larger value for the iteration variable that was mentioned earlier.
9.0 Developer’s Manual

9.1 Inventory of all figures

![Figure 17: Developing Environment for scraper](image)

Figure 17 is of the development environment used to create the data scraper. The application is called Google Apps Script and is accessible from the extensions tab of a Google Sheet. As seen above, the application provides an area for creating and running scripts, and an execution log will pop up on the bottom third of the screen with any error messages or execution details when the script is run. More information is available in the “User Manual” section.
9.2 Inventory of all relevant files

An inventory of all data files and program files is given in Table 1.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube_Data.gsheet</td>
<td>Google Sheet with web-scraping data over three sheets for each of the client’s chosen games.</td>
<td>Google Sheets spreadsheet populated by our Google script. Easily portable to .CSV for use in data visualization</td>
</tr>
<tr>
<td>GPE_Scraper.gs</td>
<td>Google Apps Script which runs the web-scraping code.</td>
<td>Scripting language which uses the YouTube API</td>
</tr>
<tr>
<td>visualizer.py</td>
<td>Python script which imports modules for data visualization, including pandas and plotly to create graphs and plots</td>
<td>Python scripting language</td>
</tr>
<tr>
<td>cleaner.py</td>
<td>Python script that filters and cleans the data</td>
<td>Python scripting language</td>
</tr>
<tr>
<td>nlp_module.py</td>
<td>Python script that takes comments as input and runs sentiment analysis</td>
<td>Python scripting language</td>
</tr>
<tr>
<td>NLP_instructions.txt</td>
<td>Text file containing detailed instructions for setting up and environment and running the above nlp_module.py</td>
<td>Text file</td>
</tr>
</tbody>
</table>

Table 1: Script, repository, and visualization files

10.0 Lessons Learned

Throughout the process of working with our client and creating our project, our team has learned many lessons. The first major thing we learned was how to efficiently manage our time and plan meetings accordingly. Our team has varying schedules, so it was imperative that we learned how to plan meetings that all of our members could attend. This would ensure that we would achieve maximum efficiency in terms of the relay of information between us and our client. Another
thing we learned was how to narrow down a list of requirements from a client and compromise on what we are able to realistically achieve. In a real-world environment, clients are often not sure of what is realistically possible and what skills the programmers may have. Luckily, our team was paired with a client that had a fairly realistic set of requirements and deliverables that were expected of us, but there were a few smaller details that we were able to refine throughout our meetings with said client. One other thing that we have learned was how to approach self-learning when it came to a topic we were mostly unfamiliar with. Before the project, our team members were not skilled in web scraping, but we were interested in and willing to learn about the subject. By coming together and watching instructional videos and reading relevant APIs, our team was able to grasp the concept and become skilled enough to create a functioning web scraper. Another concept that our team was able to understand through self-learning was the visualization of data. Through the use of instructional videos and online tutorials, our team was able to learn how to create interactive visualizations that helped make the web scraping data simple to understand.

10.1 Timeline/Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
<th>Actions</th>
</tr>
</thead>
</table>
| February 14 | Reassess Milestones for Changes Specified by the Client | • This milestone involves having a sit-down type meeting with the client in order to clear up project requirements and specifications.  
• This meeting also involves establishing regular meeting times with the client.  
• If the client mentions any new requirements or specifications, they shall be factored into the milestones. |
| February 28 | Hone Web Scraping Skills to Begin Data Scraping | • Experiment with the YouTube API in Google Scripts in order to better understand the API for final data collection.  
• This milestone also involves familiarizing ourselves with the concept of data scraping as a whole. |
| March 14   | Finish Data Scraping                           | • All YouTube data relating to military simulation games has been scraped from YouTube and has been cleaned for visualization. |
| March 31   | Analyze Dataset and Decide on Visualizations   | • Examine the data collected and decide on what approach the client wants to take in terms of how to visualize the data. |
| April 12   | Begin User Manual, Final                       | • Begin a framework for the final presentation that                         |
10.2 Problems

The main problem we initially faced was the collection of military simulation gaming video data. In order to collect the relevant data, our team utilized Google Sheets and the apps script extension that allows users to create scripts to modify said spreadsheets. The main problem with our initial approach was that the first draft of code we had written was only adding fifty entries to the Google Sheet. After a bit of initial research, we identified the problem behind the code being the fact that the scraper would only scrape data from each page of results, with each page containing only fifty results. In order to solve this problem, our team had to find a way to make the scraper move over to the next page once all the data from the page had been scraped. Another problem we faced with the data scraper was collecting relevant data in terms of region and language. Initially, the scraper was collecting video data for videos around the world and in numerous different languages. In order to solve this problem, our team also had to find a way to specify video language and region of origin.

Another problem our team faced was the cleaning of the data and the creation of the visualizations. The data from the web scraper contained a fair amount of duplicate entries, which caused problems when it came to the integrity of any visualizations made. There was also a problem where the data would be out of order in terms of publishing dates, which made the visualizations very messy and impossible to understand. The line graph specifically gave our team a small challenge due to the fact that the data points would sometimes not have lines connecting them, resulting in more of a segmented line graph.
10.3 Solutions

In order to solve the aforementioned scraping problems, our team first started by researching the YouTube API that we chose to work with. After more research, we were able to come up with a solution to the problem concerning the output of only fifty entries, by creating a for loop that would run through as many pages of results as our team specified. The loop would take an integer we supplied and provide a number of entries equivalent to fifty times said number. In order to solve the language and region of origin issue, we were able to identify another aspect of the YouTube API that allowed us to filter out any videos that were not in English and uploaded in the US. This filtering was also done in the aforementioned for loop.

In order to solve the problem of cleaning the data, the pandas, csv, and datetime libraries were applied to our cleaning Python script. The script would then take two strings for input: one for the input file name and one for the output file name. Once the user had entered their inputs, the script would read the input CSV file, remove the entries with dates before 2013, remove all the duplicates based on the unique ‘Video ID’ column, and create a new CSV file with the title specified by the user in the second request for input. In order to solve the visualization problem, our team was able to utilize the plotly library to add a type of figure called a ‘go figure’. This type of figure allowed us to add traces, representations of the data, for each of the CSVs to the same go figure. Once all the traces were added, each game would then have its own line on the line graph. Our team also solved the problem of segmented lines in the line graphs by specifying that the go figure should connect the gaps.

10.4 Future Work

Our client has proposed future work that involves retrieving groups of comments from videos within specified gaming communities, and then using natural language processing techniques such as sentiment analysis to identify and quantify subjective information in the comments, in order to determine the overall sentiment towards a particular topic or entity within the community. Our group initiated the process by developing a Python script that utilized Spacy [14], an open-source library for natural language processing, to demonstrate a proof of concept with a trained sentiment analysis model. Our client still wants to experiment with other models and sentiment analysis tools before coming to a decision on which to use for his research. To that end, future contributors will need to find and implement other natural language processing and sentiment analysis tools. Once a decision has been made, work must also be done to adapt any proof of concept into a fully developed research tool for the G.A.M.E.R. Lab to use for future Capstone Projects or research.
11.0 Acknowledgements

Dr James Ivory (jivory@vt.edu), our main client for the project, has been very helpful and communicative throughout the project process and has provided us with a great learning experience.

Dr. Edward Fox (fox@vt.edu), our professor for CS 4624, has been integral to pairing us with a project and putting us in contact with our client. Dr. Fox has also provided the class with numerous tools to track our work and set goals in order to become more efficient.

Michael Senters (michaels22@vt.edu), the graduate student our team is working with, has been essential in providing us with the details of the required portions of the project. Michael has also been very helpful in terms of meeting with us and providing assistance when needed.

Grant: Research Leave Augmentation from VT Institute for Creativity, Arts, and Technology (ICAT), Spring 2023

12.0 References


13.0 Appendices

13.1 Appendix A: Methodology Assignment

In this section, we outline the objectives of the different user roles that our system is designed to support. These roles are crucial for the effective functioning of the system and for achieving its overall goals. The users and their respective goals are as follows:

Users:

- Data Collector:
  - The primary goal of a Data Collector is to use our code-based solution for extracting metadata from military-simulation (mil-sim) YouTube videos created over the past decade. This will help gather crucial information to study the relationship between video games, current events, and political extremism.

- Data Analyzer:
  - The main objective of a Data Analyzer is to employ various analytical techniques that we have compiled in order to identify patterns and correlations between video games, current events, and political extremism. By analyzing the data, they can provide valuable insights into the dynamics of these relationships.

- Data Visualizer:
  - The primary responsibility of a Data Visualizer is to create visual representations of the data, making the findings more accessible and understandable to a wider audience. This can be achieved through various mediums, such as research papers, websites, or presentations, ensuring that the information is effectively communicated to the public.

In order to achieve our project goals, we broke them down into units of tasks and subtasks, which allowed us to systematically approach each aspect of the project. Here is an explanation of our task breakdown:

Subtasks:

- Collect Data
  - Plan Data Breadth
    - Decide data date range
      - 2013 - 2023
    - Decide view floor
    - Decide games
      - Arma 2 & 3
- Rising Storm 2: Vietnam
- War of Rights

- Write Python Code
  - Review python basics
  - Review YouTube API
    - Learn functionalities
  - Create skeleton
  - Add on to skeleton weekly
  - Finalize Python code

- Scrape Data:
  - Pull mil-sim YouTube video data in order to extract the necessary information
    - Extract video titles
    - Extract like counts
    - Extract view counts
    - Extract number of comments
    - Extract creator name
    - Extract tags if possible
      - Demonetization tag
      - Age-restriction tag

- Analyze Data
  - Data cleaning:
    - Removing and correcting errors
    - Recognizing inconsistencies
    - Acknowledging missing data values
  - Data Integration
    - Combining data from the sources we gathered
    - Formatting all data into CSV or Excel format
  - Trend Recognition
    - Identify trends in data
    - Or lack thereof

- Visualize Data
  - Create data visualizations
    - Represent trend over time
      - Or lack thereof
    - Ensure comprehension
      - Simplify data terms
      - Declutter visualizations
  - Final Report
    - Create rough draft
- Add additional information as time progresses
- Review draft
  - Correct typos
  - Clear confusing wording
- Finalize draft
  - Final Presentation
    - Create rough draft
    - Tweak rough draft over time
      - Add info over the semester
      - Make changes according to class feedback
      - Add visuals
    - Review draft
      - Correct typos
      - Review information for clarity
    - Finalize presentation
- Goal 1: Collect Data

![Collection Data Diagram](image)

Figure 18: Collection Data Diagram

- Goal 2: Analyze Data
With these goals in mind, we can plan out the necessary components of each step along the way. The best way to represent the organization of the process is to fill the steps into a comprehensive table. In the table below, we can see which services perform what task, require which files, output which files, and require which libraries, functions, and environments.
<table>
<thead>
<tr>
<th>Service ID</th>
<th>Service Name</th>
<th>Input file name(s)</th>
<th>Output file name</th>
<th>Libraries; Functions; Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Video Titles</td>
<td>Collect Data</td>
<td>N/A - Scraped from YouTube</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Extract Like Counts</td>
<td>Collect Data</td>
<td>N/A - Scraped from YouTube</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Extract View Counts</td>
<td>Collect Data</td>
<td>N/A - Scraped from YouTube</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Extract Number of Comments</td>
<td>Collect Data</td>
<td>N/A - Scraped from YouTube</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Extract Creator Names</td>
<td>Collect Data</td>
<td>N/A - Scraped from YouTube</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Extract Video Tags</td>
<td>Collect Data</td>
<td>N/A - Scraped from YouTube</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Clean Data</td>
<td>Analyze Data</td>
<td>youtube_metadata.csv</td>
<td>youtube_metadata.csv</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Trend Recognition</td>
<td>Analyze Data</td>
<td>youtube_metadata.csv</td>
<td>analysis.txt</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Data Visualization Creation</td>
<td>Visualize Data</td>
<td>youtube_metadata.csv</td>
<td>visualization1.jpeg, visualization2.jpeg, visualization3.jpeg, visualization4.jpeg</td>
<td>Google Sheet, environment of storage, Google App Script for scraping data from YouTube, and Pandas for visualizing the scraped data</td>
</tr>
<tr>
<td>Final Presentation</td>
<td>Visualize Data</td>
<td>N/A</td>
<td>presentation.pptx or presentation.mp4</td>
<td>Google Slides for the presentation development environment in the event of a PowerPoint</td>
</tr>
<tr>
<td>Final Report</td>
<td>Visualize Data</td>
<td>N/A</td>
<td>final_report.docx</td>
<td>Google Docs as the editing software for the report (exporting final result as a .docx and PDF file)</td>
</tr>
</tbody>
</table>

Table 3: Services Table

When implementing our solution, we have organized the tasks into a set of workflows that correspond to each goal. This approach ensures that the project is structured and easy to follow, as well as providing a clear path to achieving the desired outcomes. Here's a breakdown of the workflows for each goal:

An example of how one can write up this for each goal is:

Goal 1: Workflow 1 = Service 1A + Service 1B + Service 1C

Goal 1 [Collect Data]: Workflow 1 = Plan Data Breadth + Write Python Code + Scrape Data

Goal 2 [Analyze Data]: Workflow 2 = Data cleaning + Data Integration + Trend Recognition + Midterm Report
Goal 3 [Visualize Data]: Workflow 3 = Create data visualizations + Final Report + Final Presentation