COMP 400: Navigating the Bias: An In-Depth Analysis of YouTube's Recommendation Algorithm

Abstract: In this study, an algorithmic methodology utilizing web scraping techniques was introduced to navigate the YouTube recommendation system. Two empirical investigations were conducted to test the hypothesis of existence of the system's inclination towards right-wing content, potentially fostering radicalization. The initial investigation employs a seed video and traces the subsequent recommended videos, spanning five recommendation layers. The subsequent investigation adopts a combination of passive and active approaches to discern any inherent biases within the recommendation algorithm in attempts to escape the echo chamber effect. To ascertain video political affiliations, a Natural Language Processing (NLP) model was trained on a diverse dataset comprising political and non-political video metadata. Upon applying the NLP model to determine political bias based on video titles and descriptions, the results suggest a more pronounced recommendation of left-wing content, challenging the hypothesis of a predominant right-wing bias.

Table of Contents

1. Introduction

In recent political and social dynamics, marked by the rise of nationalist movements originating from online forums and the widespread circulation of conspiracy theories through digital channels, YouTube has emerged as a particularly salient platform for critique and analysis. This prominence is not solely because of the vast viewership it commands but also due to allegations regarding its potential facilitation of radical content. The platform's automated video recommendation system has been spotlighted for its purported tendency to guide users down paths leading to increasingly extremist content, thus potentially highlighting existing biases or creating new ones.

A notable 2018 article from The New York Times delved deeply into this concern, suggesting a systemic bias in YouTube's recommendation algorithm towards material that can be categorized as incendiary or provocative. The article elaborated on an observation where users, even those who began their viewing journey with relatively mainstream news or benign content, found themselves quickly pushed towards far-right or extreme viewpoints. The article proposed that this was not merely an oversight or unintended consequence. Instead, it was a calculated algorithmic strategy, intentionally exploiting the inherent human tendency for provocative content, and our deep-seated curiosity to further investigate subjects that elicit strong reactions or challenge our preconceived notions[1].

This research had two objectives. The first aim was to scrutinize the nature of content recommended by YouTube's algorithm after the initial viewing of politically-oriented videos, precisely ascertaining if consumption of right-wing material led users toward progressively extremist content. The second objective was to observe how difficult it is to escape the echo chamber of political views. This aspect was crucial for understanding whether YouTube's recommendation algorithm demonstrates a preferential bias towards right-wing viewpoints over other ideological stances.

To achieve the first goal, a web-scraping instrument designed to trace the trajectory of recommended videos by the platform's algorithm was developed. After sufficient iterations and collection of recommendation trees originating from both right- and left-wing videos, the scraper analyzed each video's political orientation. For this, a Natural Language Processing (NLP) model was utilized that was trained on a dataset comprising 2,400 videos spanning right-wing, left-wing, and non-political categories. Based on this model, the objective was to predict the political affiliation of each recommended video. Both the NLP model and the dataset employed for training are available for review in the corresponding GitHub repository where this report is hosted.

The second objective involved establishing a viewing history with pronounced ideological bias, followed by an evaluation of the home page recommendations and the use of occasional active searches in an attempt to diversify the political views. After watching 80 videos that aligned with an account's pre-determined political bias, 10 videos that opposed that bias, and 10 non-political videos, the account was considered established. An algorithm would then return to the home page and scan the top 25 recommendations for an opposing view. If no such video was found after 3 refreshes, the algorithm would actively search for a video that challenges the viewpoint.

2. Background and Related Work

Two key studies were consulted for this project. The first, titled "Algorithmic Extremism: Examining YouTube's Rabbit Hole of Radicalization" by Ledwich and Zaitsev, focused on scrutinizing 816 channels boasting over 10,000 subscribers each. The study zeroed in on channels where at least 30% of the material was political in nature. Each channel was tagged using an 18-tag system. The principal goal was to analyze how algorithmic traffic moved between these channels, aiming to discern whether YouTube's recommendation algorithm actively propelled users towards radicalized content.

Contrary to popular belief, Ledwich and Zaitsev found that YouTube's recommendation system actually deterred viewers from accessing content that endorsed radicalization or extremism. Additionally, their data analysis yielded a noteworthy pattern concerning the algorithmic preferences. The research suggested that mainstream media and cable news outlets received a favorable bias compared to independent YouTube channels, especially those with a left-leaning orientation. Specifically, they found that when a user engaged with a video from a content creator classified as Partisan Left, the algorithm delivered approximately 3.4 million more impressions to the Center/Left category than in the opposite scenario, where content from creators labeled as Partisan Right was viewed[2].

The second study examined was led by Joseph Vybihal, and Mika Desblancs and concentrated on the potential compartmentalizing effects of an automated algorithm that adheres to a recommendation system. The study aimed to explore possible right-wing bias and the promotion of radical content. Utilizing a web scraper, they initiated the experiment with a 'seed' video identified as either partisan right or partisan left, based on channel tags from the dataset provided by Ledwich and Zaitsev. The scraper then followed the algorithmic recommendations provided by YouTube, traversing five hierarchical layers and selecting three recommended videos at each level. Metadata for each video was captured and analyzed to categorize the content as either right-wing, center, or left-wing in terms of political orientation. Contrary to expectations, the study's outcomes revealed that YouTube's recommendation algorithm demonstrated a more frequent favorability towards left-wing content[3].

A significant limitation in the methodology employed by Ledwich and Zaitsev lies in their lack of requirement to log in to an account prior to constructing their list of recommended videos. While they argue that the recommendations for anonymous users would not substantially differ from those for logged-in users, this assertion is not substantiated by empirical evidence. Additionally, their research design emphasizes individual video recommendations, neglecting the recommendation patterns that emerge for users with established viewing

histories or interactions with previously recommended videos. This oversight is critical, as YouTube's recommendation algorithm is designed to adapt its suggestions based on comprehensive user profiles, rather than isolated video interactions as investigated in Module 4 of COMP480.

In contrast, the approach employed in this study aligns more closely with the methodology of Vybihal and Desblancs, wherein the scraper logs into an account for each iteration of the experiment. This allows us to more accurately simulate a user's experience, as YouTube's recommendation algorithm will generate suggestions informed by both the user profile and their video viewing history.

Conversely, a notable limitation in Desblanc's methodology is the deletion of user history between scraper runs. Desblanc argued that YouTube's algorithm tailors its recommendations based on a cumulative user profile. By clearing the history after each run of the scraper, the approach might overlook how the algorithm can offer more personalized recommendations over time. This could also miss how the recommendations might become more focused or extreme as you watch more videos.

Additionally, this study modified the operational parameters concerning video watching duration. Desblanc's scraper was configured to limit each video view to three minutes, irrespective of the video's actual length. To replicate a more authentic user behavior, an adjustment to viewing 60% of each video, with a capped limit of 10 minutes was made.

This study further enhanced the experimental design by opting to subscribe to each channel from which a video was viewed. This change aimed to incorporate an additional layer of user engagement, a factor that YouTube's algorithm considers in generating personalized recommendations. Adding measurements of user engagement like subscriptions opens up new opportunities to explore how YouTube's recommendation system really works.

3. Experimental Design

The experimental framework will be partitioned into two distinct components. Initially, the focus will be on detailing the architecture of the web scraper employed for exploring the mechanics of YouTube's recommendation algorithm, specifically in the context of algorithmic recommendations. Subsequently, the methodology designed to release users from entrenched recommendation biases or algorithmic 'holes' will be explained. This latter component will examine a combination of active and passive strategies for mitigating biases, including the intentional viewing of content with opposing political affiliations to evaluate their impact on the recommendation engine.

1) Web Scraper

a. High-level Design

In the design of the initial experimental component, a web scraper was developed specifically tailored for YouTube interaction. Upon initiation, the scraper opened a new Chrome window, navigated to YouTube's homepage, and logged into a predefined account. A 'seed' video was then inputted, selected from a dataset that was generated and that was utilized by a Natural Language Processing (NLP) model to categorize the video as either right-wing or left-wing. For each seed video, the scraper captured primary attributes and the top three algorithmically recommended videos, organizing them hierarchically in a tree data structure, with the recommendations serving as child nodes to the preceding video. This was iteratively conducted until the tree reached a depth of five layers.

Additionally, the scraper subscribed to each channel from which a video was sourced to simulate human-user behavior more accurately. Post data collection, each recommended video's primary features—most notably its title and description—were input into the NLP model for political categorization. Finally, a frequency analysis was

conducted to examine the proportion of videos tagged with either right-wing or left-wing identifiers.

b. Design Details

Logging-in

Previous research opted for Yandex's free email services instead of Google's due to challenges encountered when attempting to access Google accounts using Selenium. The latter often triggered bot detection mechanisms. Furthermore, to mitigate this issue, an additional approach was implemented where email addresses and passwords were sent character by character, each with a random delay following a normal distribution with a mean of 0.1 seconds and a standard deviation of 0.3 seconds.

Over time, it became evident that both Yandex and Google had improved their ability to detect such activities, rendering the previous methodology ineffective. Thus, a new strategy was required to address this issue.

Initially, upon entering the login name, the website prompted for captcha input. Despite experiments with an auto-clicker program designed to bypass the "I am not a robot" verification, it often required manual intervention due to the complexity of certain captchas. This manual intervention was something to be avoided, in order for a potential scaling up of the experiments. However, even after successfully bypassing simpler captchas, a recurring problem emerged. When launching any web browser, not limited to Google Chrome, using Selenium, it triggered a security alert stating "app may not be secure." (Appendix A Figure 1)

To circumvent this issue, an undetected version of the Chrome driver was used that appeared to evade all of Google's implemented security checks. Upon implementing this driver, it was observed that the library was strong enough on its own, even making the need for character-by-character input obsolete.

Seed Video

The web scraping algorithm commences with an initial 'seed' video, serving as the primary focal point for data extraction. The URL and title of this seed video are accessible by iterating through the root nodes of all tree structures located in the data folder within the repository. Political categorization of these videos was achieved by utilizing a specialized NLP model, which will be elaborated upon subsequently. Thus, each video was tagged with an appropriate political affiliation, identified as left-wing, right-wing, or non-political.

Video features

The web scraper was configured to extract a range of metadata for each video, including the title, content creator, description, number of views, likes, and video duration. For the scraper to approximate user behavior more closely, 60% of each video was viewed, subject to a maximum limit of 10 minutes. This approach was adopted to recognize the recommendation algorithm's sensitivity to user engagement time on individual videos. While viewing videos in their entirety was impractical due to time constraints, a maximum limit of 10 minutes was implemented to strike an optimal balance between the time expended in the scraping process and the simulation of genuine user interest.

Recommended Videos

In alignment with Desblanc's methodology, the decision was made to capture the first three algorithmically recommended videos to construct a multi-layered tree of subsequent recommendations. The objective was to traverse the depths of the recommendation algorithm to scrutinize the trajectory on which the scraper was directed. While the ideal scenario would have involved the creation of an extensive tree with multiple nodes and branching layers, time constraints prevented this approach.

The number of videos viewed can be modeled as a geometric series. Using $a_1 = 1$ as the first term, r as the common ratio corresponding to the number of child nodes, and

$n = tree\,depth + 1$ as the number of terms, the scraping time t for a single traversal pass can be established. This is given by $t = \frac{S_n}{2} \times 10$, where S_n represents the total number of videos viewed, 2 signifies the batch size for simultaneous video viewing, and 10 represents the maximum time allocation per video. The resulting metrics are as follows:

To find the number of nodes the sum S of the first n terms of a geometric series can be used, and the formula is the following: $S_n = \frac{a_1 \times (r^n - 1)}{r - 1}$

For $r = 3,\ n = 7 \rightarrow S_6 = 364\ and\ t = 1862\ minutes\ 30.33\ hours$

In order to optimize both computational efficiency and the depth of exploration within the recommendation system, a tree structure was utilized with a predefined depth of 5 and each node having three recommended video children. This approach balanced the need for a manageable number of tree traversals against the requirement for a comprehensive analysis of YouTube's recommendation algorithm.

Natural-Language-Processing Model (NLP)

A comprehensive dataset comprising 2400 entries was gathered with the use of YouTube API. Among these entries, a balanced distribution was maintained, encompassing 600 videos sourced from left-leaning channels, 600 from right-leaning channels, and a further 1200 originating from non-political channels. The collection process involved the utilization of the YouTube API to retrieve essential metadata including channel names, video titles, and descriptions and yielded 1200 entries that aligned with political themes.

To maintain balance, it was decided to have an equivalent quantity of non-political videos. However, due to the inherent diversity of non-political content, various genres were sampled, spanning fitness, culinary arts, comedy, education, science, travel, and entertainment, among others.

Multiple classical machine learning models such as Logistical Regression, Support Vector

Machines (SVM), Random Forest, and Naive Bayes were trained and tested on the dataset with labels being Left/Right/Non-political and features being Titles and Descriptions. Among this list, SVM model performed best, with accuracy being quite high even without hyperparameter tweaking. In order not to overfit the model to the test set with hyperparameters, the dataset was split into training, test, and validation sets. After the model was trained on titles and descriptions of videos, it was tested on both test and validation sets with metrics such as precision, recall, f1-score, support, and accuracy being high and close to each other.

This model was later used in both experiments to determine the political affiliation of YouTube videos based on their titles and descriptions.

2) Home Page Model

a) High-Level Design

The experimental procedure was to establish an account bias, either leaning towards the left or right, using results of the survey conducted in the chapter titled "Analysis of the Impact of Algorithms on User Segmentation: A Special Focus on YouTube[3]." Each account was assigned a distinct role—Left or Right—corresponding to the predominant type of videos viewed. To cultivate the bias, an account engaged with a series of 100 videos majority of which aligned with its assigned point of view.

Subsequently, the account navigated back to the homepage and analyzed the top 25 recommended videos, typically fewer due to the presence of advertisements that took place away from the videos. The objective was to identify videos presenting an opposing viewpoint (left-leaning for a right-bias account, and vice versa). If no such contrary-content videos were present, an algorithm would refresh the page up to three times. During these refreshes, if a video representing the opposing perspective emerged, the algorithm would watch the video, refresh the page, and reiterate the aforementioned steps.

In instances where opposing view videos were not found even after three refreshes, the algorithm would use the dataset of videos used in training of the NLP model. From this dataset, the algorithm would look for a video title associated with the opposing perspective and actively search for it. Upon searching and viewing the identified video, the account would return to the homepage, repeating the above steps.

b) Design Details

Logging-in

The process of logging-in was similar in both experiments

Establishing bias

Following the comprehensive survey detailed in the chapter titled "Analysis of Algorithmic Influence on User Categorization: A Special Examination of YouTube[3]," an account systematically viewed a total of 100 videos. Within this corpus, 80 videos were deliberately curated to align with the prevailing political inclination of the account, while an additional 10 videos were intentionally selected to represent opposing viewpoints, and a further 10 videos were designated as non-political, drawn from the shared dataset utilized across both experimental studies.

This approach created an obvious echo chamber effect within the account's viewing history, reinforcing its preexisting inclinations. However, the inclusion of opposing view videos and non-political content introduced an element of diversity that allowed for the presence of alternative perspectives within the account's viewing patterns, making it less artificial while still being quite radical.

Home page recommendations

An analysis was conducted on top 25 recommended videos from the home page, although it should be noted that, in a majority of instances, certain video slots were occupied by advertisements, resulting in an average count of approximately 23 to 24 videos per page. To collect a more numerous set of videos, page scrolling was required. However, the act of

scrolling posed its own set of challenges, mainly concerning inconsistent video loading times and lag after reaching the 50th recommendation.

Consequently, there were two possible solutions: first, introducing a wait time of around 5 to 10 seconds to accommodate loading and lag, and second, using the first 25 videos. Given the considerable duration already associated with the experiment, the decision was made to retain the 25-video constraint, optimizing the balance between data collection and experimental efficiency. In addition to this, the first 25 recommendations are hypothesized to be more important and closely related to the account preferences.

4.Implementation

Libraries used:

Selenium[4]:

Selenium serves as a browser automation framework. Employing Google's Chromedriver as the underlying engine, Selenium enables automated web navigation, element interaction, and item identification for subsequent data extraction by the scraping algorithm. After setting up the web driver, Selenium's main features in the following steps of scraping process were:

- **Element Identification:** The scraping algorithm employed the methods `webdriver.find_element_by_xpath(str_path)` and `webdriver.find_element_by_css_selector(str_path)` to localize web elements based on their XPath and CSS selectors, respectively.
- **Wait Mechanism:** To mitigate the risk of runtime exceptions such as `ElementNotFoundException`, the scraper incorporated an implicit wait strategy using `WebDriverWait(driver, 15).until(EC.element_to_be_clickable((By.PATH_TYPE, str_path)))`. This configuration allows a user-specified time window (in this case, 15 seconds) for certain conditions to be fulfilled, thereby ensuring element availability before

interaction. The condition specified in this context is the element's 'clickability', although other conditions such as `presence_of_element` can also be used.

Undetected-Chromedriver[5]:

The undetected-Chromedriver is a modified version of the standard Selenium Chrome WebDriver used for web automation. Its primary purpose is to bypass website mechanisms that detect and block automated activity, making the actions appear as if a human performs them, which is very useful for scrapping.

Pytube[6]

The Pytube library serves as a Python-based interface for the automated retrieval of multimedia and metadata from YouTube's platform, including individual videos and playlists. Employing a simple yet robust API enables streamlined access to video and audio streams and associated metadata. The Pytube library is principally utilized in web scraping applications to instantiate a YouTube object. This object facilitates directly extracting specific attributes such as video duration and title. For example, the video length can be obtained through the expression YouTube(video_url).length, while the title is accessible via YouTube(video_url).title. The scrapper used this information to indicate the running time for each video as well as to store the name of the video in the data structure.

pandas[7]

Pandas is a library that provides data structures and functionality for efficient work with data structures. It was primarily used for datasets and data cleanup.

sklearn[8]

Scikit-learn is a machine learning library that provides a comprehensive set of tools and algorithms for various machine learning tasks. The main purpose of this library was:

- **Splitting datasets into training/test/validation sets** using train_test_split
- **Tokenizing the collection of text documents into TF-IDF matrix** for a numerical representation of the text data using TfidfVectorizer

- **Training the NLP model on the dataset** using svm function
- **Printing model performance metrics** using classification_report and confusion_matrix functions.

Information storage for the YouTube Scrapper

For the data structures to store data, a class designated as TreeNode has been created to encapsulate the properties and behaviors of a node within a hierarchical tree architecture. Each instance of this class retains a unique identifier termed video_id and maintains a collection, specifically a list, of its immediate descendants, denoted as children. Optionally, an instance may contain a reference to its antecedent node, captured by the attribute parent. The class has the following methods: the __init__ constructor, responsible for the instantiation process, initialize each object with a specified video_id and an optional parent attribute; the add_child method is designed to extend the children list by appending new child nodes; the to_dict method recursively generates a dictionary-based representation of a node and its associated subtree; and a specialized __repr__ method that yields a textual representation of the node. Notably, the __repr__ method incorporates indentation as a visual heuristic to show the hierarchical level of each node within the overarching tree structure.This class makes it easy to build a tree-like structure to show how videos are related to each other. Each video, represented by a node, can have any number of child videos connected to it, helping you understand the relationships between them. The data structure is saved as a txt file after each scrapper run.

Process of the YouTube Scrapper

Initialization Phase:

In the initial stage of the web scraping algorithm, an automated login to a YouTube account is facilitated via the Undetected Chrome WebDriver. Subsequently, the scraper inputs a URL link corresponding to a YouTube video extracted from a predefined list comprising categorically sorted seed videos. This list serves as the source for root nodes to be traversed by the

algorithm during each execution cycle, i.e. the seed videos. A new scraper object is instantiated for every individual run, employing one of the seed videos as its initialization parameter. Upon successful selection of the seed video, the scraper commences the process of video stream interaction, effectively simulating the act of viewing the video.

Operational Phase:

Upon invocation of the `main()` function, the scraper initializes a Chrome browser window, employing specific WebDriver options to mute audio output and deactivate the blink feature. Following this, the scraper navigates to the YouTube homepage and logs in using the provided login information. For the scraper to mitigate the detection of automated bot activity, temporal delays are programmatically incorporated between actions, replicating human-like interaction via keyboard and mouse inputs. Subsequently, the scraper invokes the watch_video(driver, seed_video) function, using a seed video as a parameter. This function is responsible for redirecting the WebDriver to the video's URL, executing subscription to the channel through subscribe_channel(driver), and waiting until the ads are skippable via wait_for_ad_to_finish(driver).

Post-advertisement, the get_duration(video_url) function is called, directing the WebDriver to engage with the video stream for a duration corresponding to 60% of the video's total length, subject to a 10-minute maximum constraint. The video stream is paused after this viewing period, and the process_related_video(driver, video_url, current_depth, max_depth, skip_first_recommendation=False) method is invoked. This method is designed to extract metadata for the top three recommended videos adjacent to the currently viewed video. Concurrently, the visited video is recorded within the tree data structure, as delineated in the preceding explanation of the TreeNode class. The recommended videos are watched in a depth-first order.

Depth Management and Data Storage:

The algorithm employs a variable to monitor the depth of recommendations traversed at each hierarchical level, ensuring that the traversal does not exceed a pre-defined max_depth of 5. Upon reaching this depth limit or exhausting the list of recommended videos, the URLs of all visited videos are saved into a txt file. This text-based representation uses hierarchical indentation to visually represent the parent-child relationships between the videos. Concurrently, a JSON file is generated to encapsulate the tree structure. In this JSON representation, dictionaries are employed to model the child nodes corresponding to each parent node, thus facilitating a structured, machine-readable tree storage.

Process of Home Page Model

1) **Establish Bias**: Watch 100 videos with the 80-10-10 split to establish a political lean. watch_video(driver, video_title, channel_name) function simulates the process of searching for and watching a YouTube video. Because titles are not unique the function searches the video by title and channel name, clicks on the matched video, skips ads if present, and watches the video. sample_videos(df, n) function samples videos from the dataframe latest_videos.csv based on their political affiliation. The function returns a shuffled dataset with the sampled videos with 80-10-10 percentage split.

2) **First Check**: Open the home page and categorize the top 25 recommended videos based on their political affiliation using the NLP model. check_and_record_recommendations(driver) function collects video recommendations from the YouTube homepage, predicts their political affiliations using an imported SVM function (predict_political_affiliation), and records the recommendations along with their affiliations. It also counts the occurrences of different political affiliations and writes these counts to a CSV file (numbers.csv).

3) **Refresh Mechanism**:

If an opposing view is not present in the initial top 25, refresh the page up to 3 times to check for any changes in recommendations.

Record the political affiliations of recommended videos after each refresh.

4) **Iterative View & Check**:

If an opposing view appears at any point (either in the initial check or any of the refreshes), watch that video. watch_recommended_video(driver, recommendations, right_video_titles, target_affiliation) watches a recommended video with a specific target political affiliation. It searches through a list of recommendations to find a video with the specified affiliation.

Return to the home page and repeat the process, starting from step 2.

5)**Active Search**:

If, after 3 refreshes, still no opposing view appears, then proceed to actively search for and watch a video with an opposing view using watch_recommended_video(driver, recommendations, right_video_titles, target_affiliation) function.

Return to the home page and repeat from step 2.

Finally, the main() function performs all these tasks in addition to logging-in, creating csv files, txt file logs, initializing ChromeOptions and Chrome browser.

5. Hypothesis evaluation & Data Analysis

In this data analysis section both model's data will be investigated

Claim: Right-Wing Advantage:

In line with popular opinion and assertions made in a New York Times article, it is claimed that YouTube's recommendation algorithm demonstrates a predisposition towards right-wing content relative to alternative viewpoints. However, this claim appears unsubstantiated upon empirical analysis of the collected data. This finding is consistent with outcomes from two other independent studies discussed earlier, thereby casting further doubt on the claim's validity that YouTube's recommendation algorithm inherently favors right-wing content.

The study counted how many recommended videos were labeled as either right-wing or left-wing, based on whether the seed video was from the right or the left. This distribution is graphically depicted in Figure1. According to the empirical data, out of a total of 717 videos recommended from the left-seed videos, 85.9% were identified as left-wing, 6.8% as right-wing, and the remaining 7.3% as non-political. Conversely, in the case of the 1,161 videos recommended following right-seed videos, only 24.1% were characterized as right-wing, while a substantial 79.8% were labeled as left-wing and 6.1% as non-political.

The higher frequency of right-seed video recommendations was intentionally incorporated into the experimental design to rigorously validate the initial findings, particularly considering the high number of left-wing video recommendations. Although the data indicates that right-wing recommendations quadrupled when initiated by right-seed videos, the predominance of left-wing recommendations remains noteworthy. One plausible explanation for this phenomenon could be the higher volume of left-wing content producers relative to their right-wing counterparts. Additionally, mainstream media outlets are frequently classified as predominantly left-wing in orientation. For the study to delve deeper into this asymmetry, it examined the first video recommended following a right-seed video initiation. Remarkably, instances were observed where the initial recommendation was sourced from left-wing channels, as shown in Appendix A Figure 2.

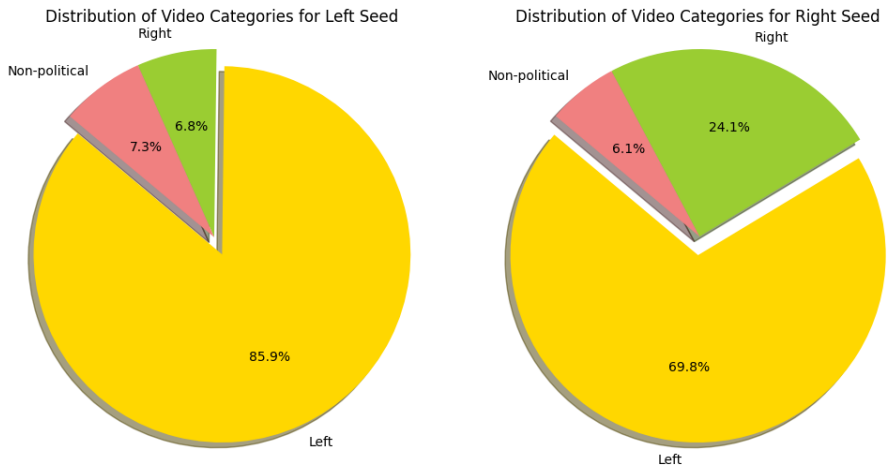

Figure 1 Distribution of Video Categories

Sequential analysis was used to understand the study's results better. This computational approach is designed to elucidate the structural dynamics underpinning the political orientation of consecutively recommended videos, categorically classified as left-leaning, right-leaning, or non-political. Utilizing state transition modeling, the sequential analysis quantifies the conditional probabilities governing transitions from one political category to another within the recommendation chain. This enables an empirical evaluation of systemic biases present within the algorithmic framework of the video recommendation engine. By implementing this analytical method, the study determined if the platform commonly recommends videos that share the political leaning of the initial video viewed. This tendency to suggest similar types of content is often called the "echo chamber" effect. In other words, if one starts with a video that leans politically in one direction, the subsequent recommendations are likely to lean in that same direction as well.

The results are as follows:

1) In the context of an initial left-seeded video, the state transition probabilities exhibit distinct patterns based on the originating political orientation. When initiating from a "Left" state, empirical evidence suggests an 86.53% probability of staying within the same ideological

category upon the following recommendations. Conversely, there is a 6.66% likelihood of transitioning to a "Right" state and a 6.82% probability of venturing into a "Non-political" state. Should the viewer find themselves in a "Right" state despite commencing with a left-seeded video, the data indicates an 87.5% probability of reverting to a "Left" state, coupled with a 6.25% chance of maintaining the "Right" state and an equivalent 6.25% likelihood of transitioning to a "Non-political" domain. Lastly, when the initial state is "Non-political," the subsequent video recommendation is most likely to be left-leaning with a 76.92% probability, followed by a 9.62% chance of transitioning to a "Right" state and a 13.46% probability of remaining in the "Non-political" realm (Figure 2).

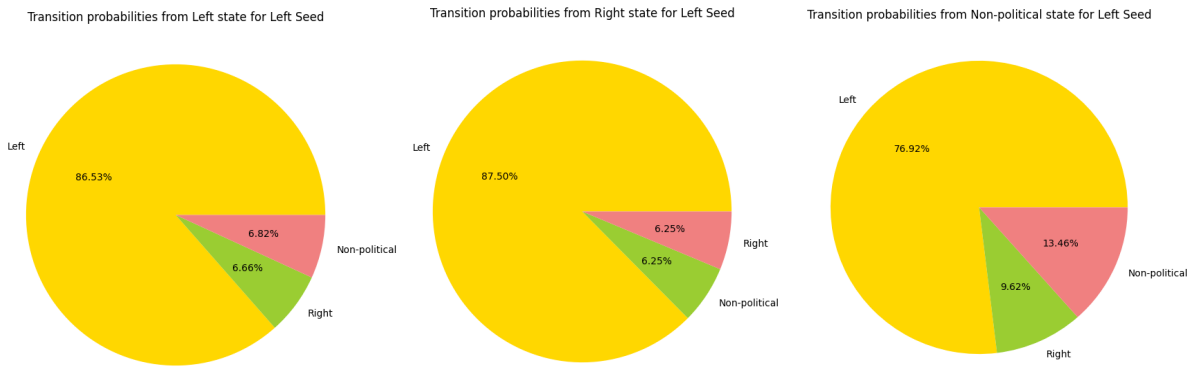

Figure 2: Transition Probabilities when starting with left seed

2) In the scenario where the video recommender is initiated with a right-seeded video, the state transition probabilities manifest specific tendencies correlated with the initial ideological orientation. When beginning from a "Left" state, the empirical model indicates a substantial likelihood of 72.44% to persist within the same ideological classification upon receiving the subsequent video recommendations. There is additionally a 21.88% probability of transitioning to a "Right" state and a more modest likelihood of 5.69% to enter a "Non-political" category.

If the initial state is "Right," statistical analysis reveals a 62.86% probability of transitioning to a "Left" state, a 34.29% chance of remaining in the same "Right" state, and a relatively minuscule 2.86% probability of transitioning into a "Non-political" domain.

Finally, when the recommender starts in a "Non-political" state, the ensuing video recommendation is most likely to be "Left" with a probability of 65.71%. This is followed by a 10% likelihood of transitioning to a "Right" state and a 24.29% chance of remaining within the "Non-political" category (Figure 3).

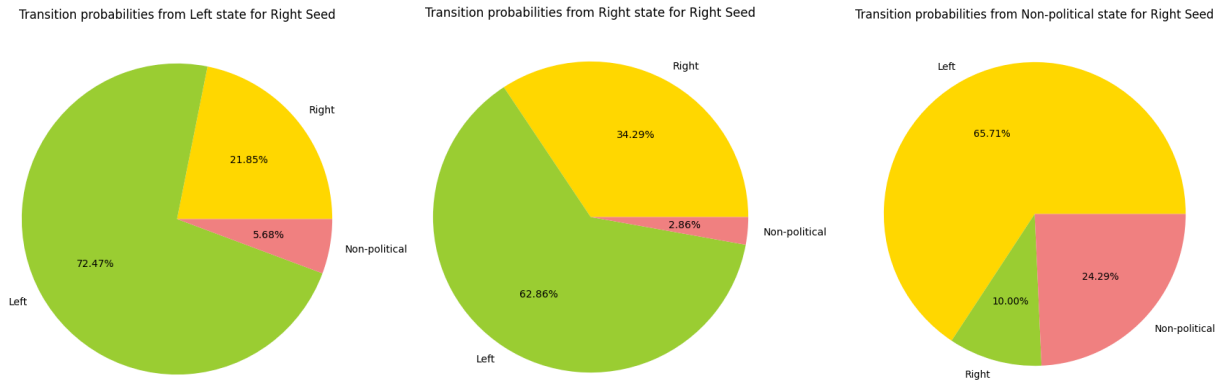

Figure 3: Transition Probabilities when starting with right seed

In the home page experiment, a left-leaning account that transitioned into a right-leaning account initially had greater left video representation on the homepage, as expected. Throughout the conversion period, the ratio of left and right videos remained relatively similar, which meant a rise in right-wing recommendations. This can be observed further with the final trend that implies the rise of right-wing videos in the home page recommendations (Figure 4).

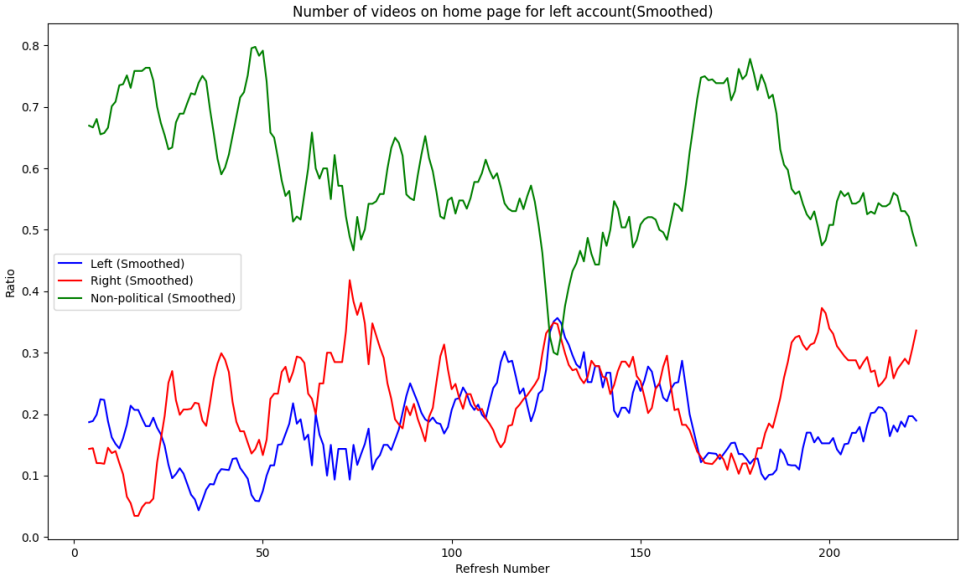

Figure 4 : Political affiliation ratio of left-wing account

For a right-leaning account, the initial conditions remained similar. After watching primarily right-wing content, the first few iterations of the algorithm revealed the dominance of right-wing content over the left-wing. However, the continuation of this trend for the first 67.8% of the iterations was drastically different. Left-wing content started to achieve the same levels of representation at the very end of the iterative cycle, and after the last 100th watched video, it

still remained tied with right-wing content.

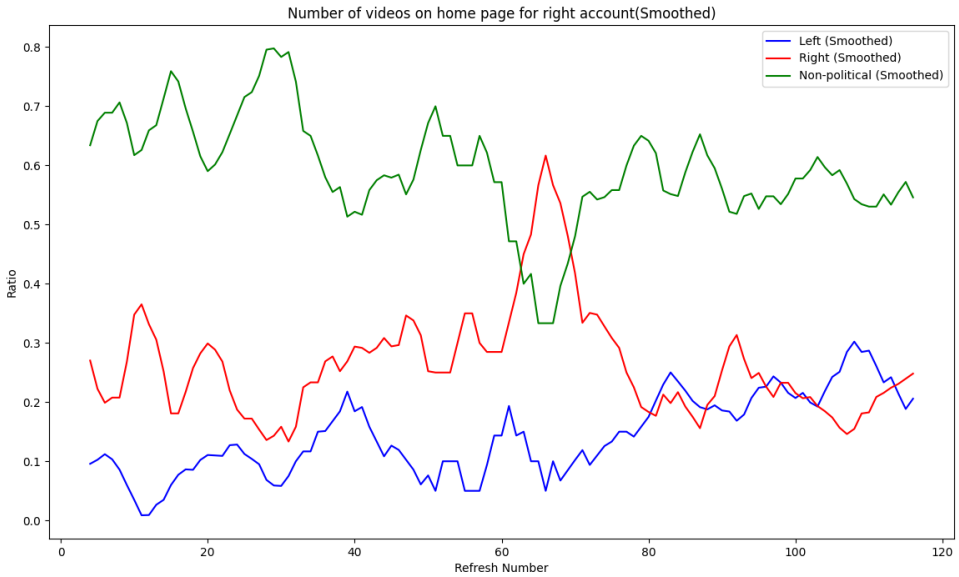

Figure 5: Political affiliation ratio of right-wing account

Another considerable difference is the average number of page refreshes needed to find an opposing view. The left-leaning account required, on average, two page refreshes in order to find a right-wing video to watch. In contrast, the right-leaning account had less difficulty finding an opposing view, averaging one refresh per video.

6. Limitations & Conclusion

Limitations

The methodology employed in this study needs to be revised to constrain its conclusions' validity. Specifically, the experiments aimed to assess the presence of right-wing bias in the YouTube recommendation algorithm, and such a bias could potentially contribute to user radicalization. For the study to investigate this claim, a web scraper was developed to simulate

user behavior by watching 60% of a YouTube video and then following the platform's recommendation algorithm. The home page recommendation bot watched a total of 10 minutes per video

However, this approach does not adequately capture the complexity of user engagement with the platform. In real-world scenarios, users select their recommended content, sometimes selecting the first suggested video, at other times choosing a video further down the recommendation list, or even searching directly for specific content. The experimental design restricted itself to the immediate three video recommendations, offering a limited representation of user interaction with the recommendation system. Moreover, the algorithm automatically subscribed to channels, a behavior that may not accurately reflect human users' more selective subscription practices. Additionally, users deeply entrenched in an echo chamber of beliefs rarely try to break free from its influence radically and forcefully. This is normally a gradual and time-consuming process that occurs over multiple sessions.

Thus, these methodological shortcomings limit the generalizability of the study's findings.

In the study, time constraints limited the depth to which the web scrapers could operate, the number of recommendations per video, and the number of home page recommendations that could be analyzed.

Conclusion

The research introduced YouTube scraper scripts, detailing their ability to bypass Google's login security measures and outlining the data collection and organization methods. The findings contradicted the assumption that YouTube's recommendation algorithm disproportionately suggests right-wing content. These results are congruent with previous studies by Ledwitch, Zaitsev, Vybihal, and Desblancs, which also investigated the YouTube recommendation system. Interestingly, the study revealed that users identified as right-wing received a significantly higher number of recommendations for left-wing videos. Conversely, left-wing users were exposed to negligible right-wing content in their recommendations. These observations suggest that

additional research is warranted to understand the YouTube recommendation system's complexities fully. Further data collection and more comprehensive temporal analysis will be the primary focus of subsequent phases of this investigation to ensure that the results remain stable and consistent throughout time and multiple sessions.

The complete codebase and relevant data structures are publicly available in the designated GitHub repository.

Appendix A:

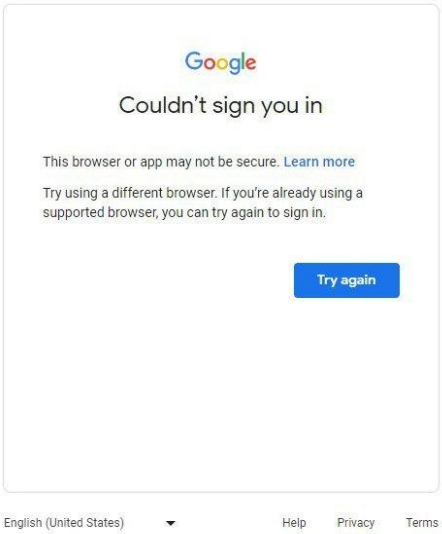

Figure 1

Figure 2

Work Cited:

1) Tufekci, Zeynep. “YouTube, the Great Radicalizer.” The New York Times, The New York Times, 10 Mar. 2018, www.nytimes.com/2018/03/10/opinion/sunday/youtube-politics-radical.html.**0**
2) Ledwich, M., & Zaitsev, A. (2020). Algorithmic extremism: Examining YouTube’s rabbit hole of radicalization. First Monday. https://doi.org/10.5210/fm.v25i3.10419
3) Mika-Jpd. “Mika-Jpd/Youtube_radicalization_recommendations: Chapter 4: Analysis of the Impact of Algorithms in Siloing Users: Special Focus on YouTube for the Book ‘Ai & Society: Tensions and Opportunities.’” *GitHub*,
4) Muthukadan, Baiju. “Selenium with Python.” Selenium with Python - Selenium Python Bindings 2 Documentation, 2011, selenium-python.readthedocs.io/index.html
5) “Undetected-Chromedriver.” *PyPI*, pypi.org/project/undetected-chromedriver/#history. Accessed 24 Aug. 2023.
6) “Pytube.” *Pytube*, pytube.io/en/latest/. Accessed 24 Aug. 2023.
7) “Pandas.” *Pandas*, pandas.pydata.org/. Accessed 24 Aug. 2023.
8) “Learn.” *Scikit*, scikit-learn.org/stable/. Accessed 24 Aug. 2023.