Faculty of Natural and Mathematical Sciences
Department of Informatics

Bush House, King's College London
Strand Campus,30 Aldwych
London WC2B 4BG
Telephone 20 7848 2145
Fax 020 7848 2851

7CCSMPRJ

Individual Project Submission 2017/18

Name:	Rounak Saha
Student Number:	1624601
Degree Programme:	MSc Advanced Computing
Project Title:	From Russia, With Love: An Analysis Of Micro-Targeted Facebook Advertisements Published By The IRA In The 2016 US Presidential Elections
Supervisor:	Dr. Sanjay Modgil
Word Count:	14,912 Words

☑ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: Rounak Saha **Date:** August 24, 2018

Department of Informatics

King's College London

United Kingdom

7CCSMPRJ Individual Project

From Russia, With Love: An Analysis Of Micro-Targeted Facebook Advertisements Published By The IRA In The 2016 US Presidential Elections

---

Name: **Rounak Saha**

Student Number: 1624601

Course: MSc Advanced Computing

Supervisor: Dr. Sanjay Modgil

This dissertation is submitted for the degree of MSc in MSc Advanced Computing.

Acknowledgements

I would like to thank my supervisor Dr. Sanjay Modgil for providing me the oppurtunity to propose this project and carry it out. I would also like to thank Dr. Nishanth Sastry for his constant support, patience and continuous feedback regarding the progress of this project.

I would extend my gratitude towards Sagar Joglekar and Pushkal Agarwal of the Department of Informatics, King's College London and the entire NetSys Research Group for continuous feedback on this project.

Finally, I would like to thank Chevening Scholarships, the UK government's global scholarship programme, funded by the Foreign and Commonwealth Office (FCO) and partner organisations for providing me the oppurtunity to pursue a Masters Degree in United Kingdom.

Abstract

Digital advertisements have transformed how advertisements are created, published and perceived by individuals. Political parties use digital advertisements to their advantage to reach voters in a personalised fashion. However, these advertisements are often published under a veil of anonymity. This enables malicious actors to reach audiences without any consequences. It was claimed that the Internet Research Agency in Russia used digital advertisements to interfere with the USA presidential elections in 2016. The U.S House of Representatives Permanent Select Committee on Intelligence investigated such claims and published 3,517 Facebook advertisements and 3,841 Twitter accounts that are linked to the Internet Research Agency (IRA) in Russia [29].

In this project, we analyse the IRA published Facebook advertisements and their corresponding campaign techniques. First, we identify the micro-targeting parameters used by these advertisements. Second, we measure the degree of influence of these advertisements. We discover that the IRA used apt targeting parameters to reach audience. However, the campaign did not resolve to have a substantial degree of influence.

This leads us to investigate why the campaign did not garner sufficient momentum. We track the campaign objectives in each state in USA. We find that the campaign did not tend to have any topical objectivity and ranged over spectrum of generic topics. We further compare the topical issues used by these advertisements with the actual topical issues of the targeted geographical regions using Google Trends. We find that the topical issues used to reach audiences do not represent the actual topical issues amongst voters.

Abbreviations

CPC	Cost Per Click
CTR	Click Through Rate
CPM	Cost Per Mile
USA	United States of America
USD	US Dollars
RUB	Russian Rubbles
IRA	Internet Research Agency
PDF	Probablity Distribution Function
CDF	Cumulative Distribution Function

Contents

List of Figures

List of Tables

1 Introduction

Door-to-door campaigning or canvassing is a campaigning technique often used by politicians to pitch themselves as a suitable candidate. The candidate visits homes of voters in the contestating area and addresses regional topical issues. This technique of campaigning enables politicians to connect with their voters at a personal level and ask for their vote.

The evolution of technology has provided us with a different form of campaigning medium - digital advertisements. Voters can now be reached using digital social networks. With the aid of digital micro-targeted advertisements, personalised content can be published to different demographics of voters. While the notion of personalised micro-targeted advertisements seems to be benign, they are actually perilous in nature.

The sphere of digital advertisements is masked under a veil of anonymity. Political parties are therefore not accountable for the advertisements they publish. Malicious actors can spread misinformation under this veil. Political groups with ulterior motives may hijack entire elections by publishing digital advertisements remotely without any accountability.

In this project, we analyse the role of state sponsored advertisements published by the Internet Research Agency in Russia to influence the presidential elections held in USA in 2016. In January 2017, the National Security Agency of The United States of America published a report which claimed that the IRA interfered with the presidential elections using social media [20]. This claim was highly important. If true, it undermined the democracy of The United States of America.

The U.S House of Representatives Permanent Select Committee on Intelligence launched an investigation to address such claims. This enforced Facebook, Twitter and Google to launch independent investigations. As a result of these investigations, The U.S House of Representatives Permanent Select Committee on Intelligence concluded that the IRA did use social media to influence the presidential elections. They further list all Facebook advertisements published by the IRA as well as all identified Twitter accounts which tried to influence opinions of voters in the election [29]. This report was published as recently as February 16, 2018.

USA 2016 elections was the first of many elections which have been claimed to be fought on social media. Germany 2017, France 2017 and BREXIT are a few other elections which are considered to have had an impact because of social media conversations. This piqued the interests of researchers to analyse political campaign techniques used on social media. In [44], the authors identify 7 groups apart from the IRA who tried to influence the presidential elections in 2016. In [61], the authors try to quantify the degree of influence IRA sponsored Twitter accounts had on the presidential election.

In this project, we focus on the advertisements published by the IRA. We find that the IRA targeted voters based on their location and a set of issues exclusive to that location. Next, we quantify the percentage of US population which may have been influenced by the advertisements published by the IRA. We discover that the degree of influence resulting from these advertisements is not sufficient enough to have influenced the entire presidential elections.

We ask ourselves, if the IRA did target advertisements to users based on their location and personal interests, why were they not able to have a greater degree of influence. To this extent, we analyse the campaigning strategies used by the IRA. We first measure the topical objectivity of the advertisements in each state. We find the campaigns did not have any topical objectivity in a majority of the states.

Finally, we measure the relevance of the topical issues used by the IRA with the actual topical issues in the targeted geographical regions. We use Google Trends to locate regional topical issues. We find that the topics used by the IRA do not represent the actual concerns of the voters. This provides us enough evidence to conclude that the campaigns drawn out by the IRA did not represent relevant concerns of the voters in the targeted geographical regions. Further, the campaigns lacked objectivity and focus. These can be considered as major reasons why the campaign failed to garner substantial momentum.

This thesis is structured as follows :

- **Chapter 2.** This Chapter introduces the reader to the platform of Facebook advertisements and Twitter.
- **Chapter 3.** This Chapter discusses the interference of IRA in the presidential elections using social media in detail. The results of the investigations by various agencies are summarised in this Chapter. This Chapter concludes by discussing the **state of the art literature** in this domain. While this project deals with influencing users on social media, inspiration can be drawn from several fields of research. Relevant literature of these research domains is presented in this Chapter.
- **Chapter 4.** This Chapter introduces the reader to the three focal datasets - Facebook advertisements, Twitter Troll Mentions and Google Trends. Relevant methods used to collect and clean the data is outlined in this Chapter.
- **Chapter 5.** This Chapter takes the reader through the various methods and theories used to carry out the experiments in this project. This Chapter also casts light on how these methods and theories are implemented in the context of our project.
- **Chapter 6.** This is the focal Chapter of the project. This Chapter takes the reader through our results and the corresponding conclusions we draw from them.
- **Chapter 7.** The final Chapter concludes the project by summarising the results and the inferences we draw from them. The key findings are reported in this Chapter. Finally, we cast light on the future work that can be pursued from the findings from this project.

2 Background

2.1 Facebook Advertisements

The first known type of advertisements on Facebook were known as “Flyers”. Flyers were textual advertisements that would appear on the right hand section of the Facebook homepage. Facebook’s database of users enabled them to reach audiences in a highly targeted fashion since day zero.

In 2007, Facebook launched Facebook Advertisements. The Facebook advertisements that we see today is a highly evolved version of this feature launched in 2007. In 2008, Facebook launched another feature known as Pages [1]. Pages enabled business and individuals to form an online presence to interact with their followers. The combination of Pages and Facebook Advertisements launched in 2007-08 were the basic building blocks of the highly evolved version of Facebook advertisements we see today. Facebook has kept adding features to its advertisement portal over time to provide marketeers more targeted access to their audiences [36][43].

Targeting Parameters. Facebook gathers its data about users from the content generated by them. This data is further used to publish targeted advertisements. Facebook logs interactions made by users with different pages, users and advertisements that appear on their news feeds. These logs are further mined to assign personality traits to users known as “Facebook Interests”. Facebook also actively tracks **user location**. Details of a user’s location log is used to serve advertisements to them. Further details on Facebook Interests and User Location can be found in Section A.1.

Facebook also keeps a record of a users gender, age as well as the device and network used to browse Facebook. A combination of these parameters are used to serve advertisements to users.

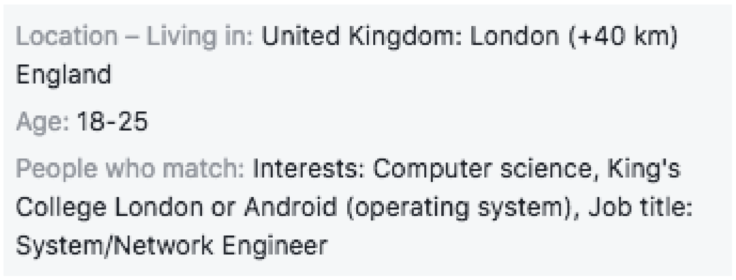
Publishing Advertisements. Facebook allows marketeers to publish advertisements using two broad functionalities : a) Boost Posts and b) Facebook Ads Manager.

Boost Posts. Posts are content that can be published by page owners on Facebook.

[1]https://www.facebook.com/business/learn/facebook-page-basics

This content can be in the form of an image, link to a website or even a video. One of the methods provided by Facebook to reach a greater number of audiences is via Boost Posts. Page owners can easily boost their post using Publishing Tools available on the Page. Page owners can also use the "Boost Post" button available to them on the bottom right hand side of every post. Perhaps, this is the most basic form of Facebook advertising available today [1].

Boost Posts can satisfy two objectives. First, one may target users that already like their page to increase engagement. Owners can also target friends of users who already like the page to increase the network of audience. Second, page owners can target new audiences to increase the user base of their page. This is done via custom targeting. In order to custom target an audience, the target gender, age, location and interests need to be provided. Facebook filters audiences according to the specified parameters and displays the advertisement to this set of users only. Figure 1 displays an advertisement that targets individuals in the age group 18 - 25 years of all genders living in London. Additionally, these individuals should also have interests such as Computer Science, King's College London, Android (Operating System) and must be interested in or have a job title of System/Network Engineer. This could be a potential advertisement for a research group in King's College London reaching out to young system engineers.

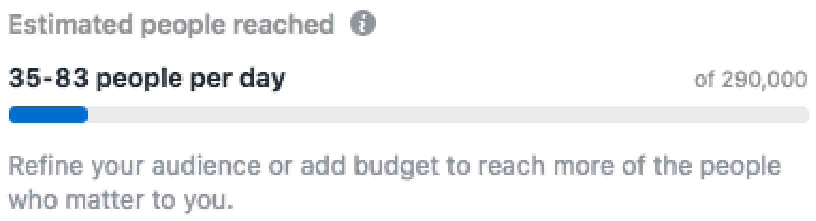

Location – Living in: United Kingdom: London (+40 km) England

Age: 18-25

People who match: Interests: Computer science, King's College London or Android (operating system), Job title: System/Network Engineer

Figure 1: Targeting Parameters Of A Facebook Boost Post

Finally, the owner of the page has to provide a budget and specify a duration for which they want to run the advertisement. Facebook in turn informs the owner about the size of the audience that can be reached corresponding to the targeting details, budget and duration of advertisement in one day. Figure 2 depicts the potential audience size as 2,90,000 for the demographics and interests mentioned in Figure 1. It also shows that 35-83 people will view the post per day at a cost of GBP 1 per day. It must be noted, page owners can run the same advertisement on Instagram as well at no extra cost.


Estimated people reached

35-83 people per day

of 290,000

Refine your audience or add budget to reach more of the people who matter to you.

Figure 2: Potential And Actual Reach Of Advertisement Depicted In Figure 1 When Boosted For GBP 1 Per Day

Boost Post can serve only one type of advertisements - Facebook Posts. This is very restrictive in terms of an advertisement campaign which may have varying content. In addition, Boost Posts only appear on the users news feeds. Advertising campaigns often require variable customised placements.

Facebook Ads Manager. Facebook Ads Manager[2] reflects the true potential of Facebook advertisements. Facebook Ads Manager builds on already available features in Boost Posts and provides a diverse set of options to build a sophisticated Ad Campaign. Page owners can publish advertisements for their pages using this portal.

First, Ad Manager provides a variety of Ad Campaigns. Every campaign focuses on a certain objective. Correspondingly, the format of the advertisement changes according to the objective. It is not limited to posts as is the case with Boost Posts. Facebook provides three broad categories of objectives :

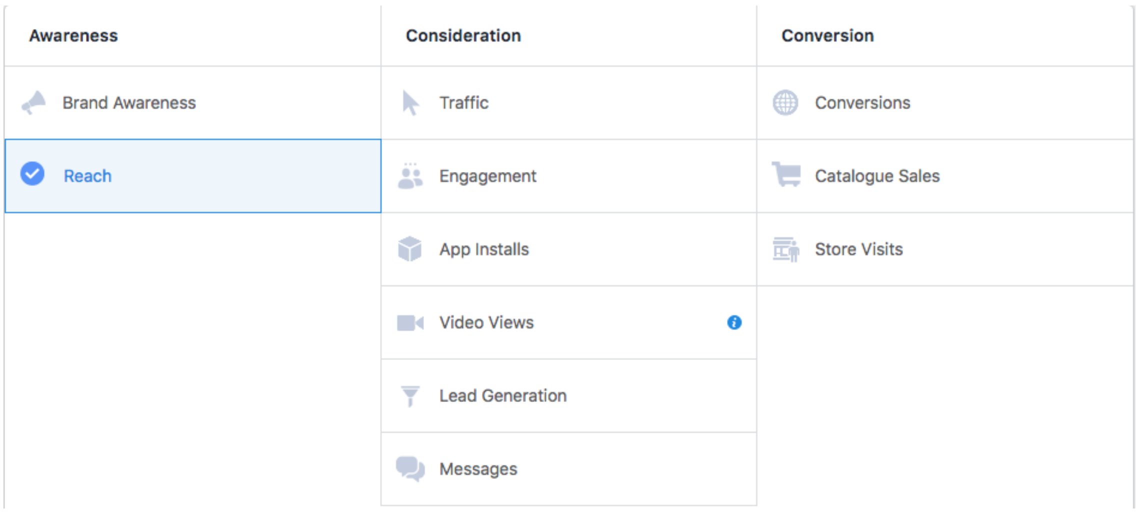
- **Awareness** - The goal is to reach a maximal number of individuals. Marketeers use this strategy to generate interest amongst their audience.

- **Consideration** - This type of advertisement campaign generates engagement amongst audiences.

[2] https://www.facebook.com/adsmanager/

- **Conversions** - Conversions lead audiences to webpages of product being sold or drive conversation about the product via Facebook Messenger.

Once an ad objective is selected, the format of the advertisement can be picked. The format is a template provided by Facebook. The format of the advertisement is directly linked with the objective of advertisement [5]. The array of objectives that one can choose from is described in Table 9. A screenshot of the objectives available on the Ad Manager platform is presented in Figure 3.



Awareness	Consideration	Conversion
Brand Awareness	Traffic	Conversions
✓ Reach	Engagement	Catalogue Sales
	App Installs	Store Visits
	Video Views	
	Lead Generation	
	Messages	

Figure 3: Screenshot of available Advertisement Campaigns on Facebook Ad Manager

Second, one can choose the targeting parameters of the advertisement. There are three broad methods to do so.

- **Standard Targeting** - Facebook Ad Manager provides the standard targeting practice where one can mention target demographics, interests, age, gender and language using a form. This is the most standard practice and is an extension upon the Boost post functionality.

- **Custom Audience** - Facebook allows marketeers to upload files of phone numbers or website visit logs. In turn, Facebook filters users based upon these logs and targets advertisements to these users only.

- **Lookalike Audience** - This feature automatically determines custom targeting parameters. Marketeers need to provide a sample of page they own or an advertisement that has performed well. Facebook regenerates a similar audience to serve the advertisements.

In addition to targeting ads to users by location, interests, age, and gender. One can also target advertisements by **language** and **connection type** on the ads manager platform. Connection Type allows marketeers to reach audiences of other pages they may own.

Finally, Facebook ads manager allows marketeers to select the placements of the advertisements. A variety of placements can be selected. Available placements are shown in Figure 4.

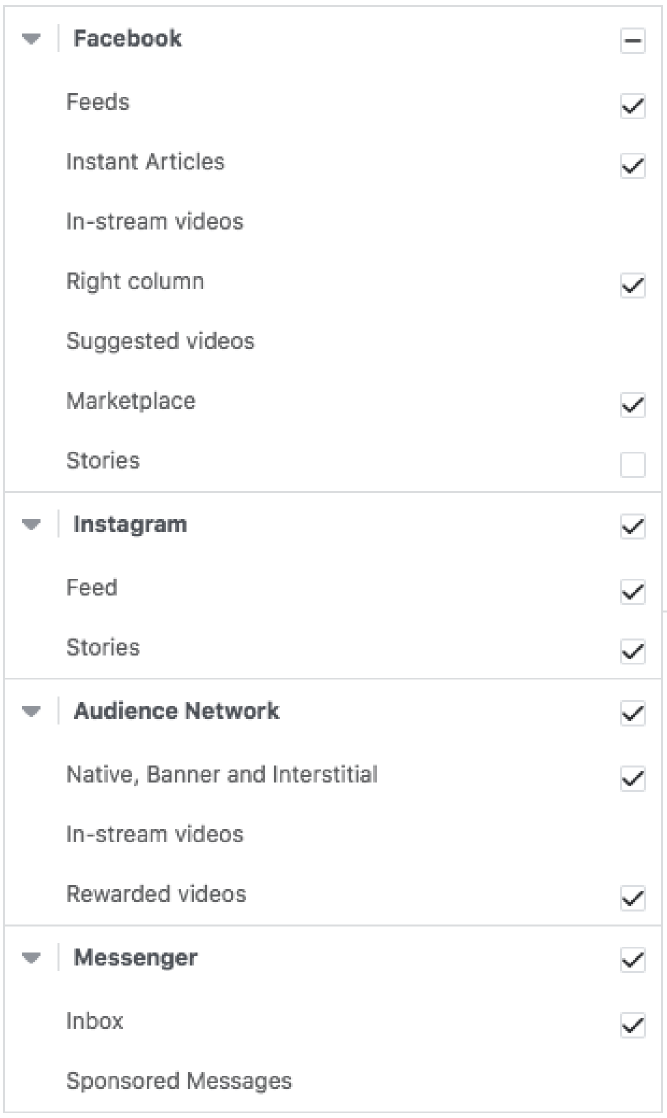
2.2 Twitter Ecosystem

Twitter allows users to publish short textual messages known as Tweets. These messages consists of 140 characters. From September 2017, Twitter increased the character limit to 480 characters [24]. Users can follow other users on Twitter. In must be noted that if User X follows User Y, it does not necessarily mean that User Y follows User X as well. Each user can be uniquely identified by their handle which is an unique string.

Users can re-publish tweets from other users. This is known as Re-Tweeting. Users can also “mention” other users in their tweets by using their handles prefixed by the @ symbol.

Twitter also verifies users based on their physical presence. Verified profiles are accompanied with a Blue Tick. This provides credibility to the profile and the network spread of such profiles are often large with a huge number of followers.

In the context of our project, we are concerned with analysing the network spread of certain accounts on Twitter known as “Troll Accounts”.



Facebook	−
Feeds	✓
Instant Articles	✓
In-stream videos	
Right column	✓
Suggested videos	
Marketplace	✓
Stories	☐
Instagram	✓
Feed	✓
Stories	✓
Audience Network	✓
Native, Banner and Interstitial	✓
In-stream videos	
Rewarded videos	✓
Messenger	✓
Inbox	✓
Sponsored Messages	

Figure 4: Availble Placements for Facebook Advertisments

3 Related Work

3.1 The Russian Influence in USA 2016 Elections

In light of state sponsored advertisement campaigns to influence the presidential election, the U.S House of Representatives Permanent Select Committee on Intelligence launched an investigation to look into such claims. In [29], the investigation committee claims, "Russian President Vladimir Putin ordered an influence campaign in 2016 aimed at the US presidential election. Russia's goals were to undermine public faith in the US democratic process, denigrate Secretary Clinton, and harm her electability and potential presidency". The investigations revealed a number of Facebook Pages that were created by the Internet Research Agency (IRA). These pages were functional from mid 2015 to mid 2017 when they were probably taken down. The pages published advertisements on Facebook whose contents are now available in the public domain. They also published around 80,000 organic pieces of content which were posts from these pages. As per [29], the content of these posts will made available in the future.

[29] claims that there were 3,841 Twitter accounts associated with the IRA which tried to influence opinions of voters. All Twitter accounts as well as Facebook and Instagram pages linked to the IRA were consecutively banned [14].

In light of such allegations, Facebook published a white paper and a blog post which highlights the efforts of the IRA to influence the election in terms of advertisements [15] [17]. To summarise the contents, Facebook believes that the advertisements published by the IRA did not directly mention the presidential elections. Therefore, they claim that the motive of the IRA was not to influence the elections. They further add that sensitive topical issues were used in these advertisements such as LGBT issues, gun rights and race which is in accordance with the analysis provided in multiple studies [50] [44].

However, in [49], the author questions the analysis provided by Facebook due to its shifting stance over time. In addition, there has been no methodology provided on how these accounts were linked to the IRA. This can also be extended to Twitter. Twitter banned a list of accounts which were claimed to be linked to the IRA. However, no explanation was provided on how Twitter determined that these accounts were actually linked to the IRA [26]. Twitter further banned users who were actively following these troll accounts.

Facebook and Twitter are two social media platforms which have been in the limelight in terms of IRA influencing the elections. Google is a tech giant which also serves as a digital advertising platform. The reach of Google extends from mail inboxes (G-Mail) to advertisements shown in between YouTube videos. This makes the reach and consequently the degree of influence from Google advertisements of much more consequence than either Facebook or Twitter [16]. However, no formal investigation results have been published with regards to Google.

3.2 Influencing Political Opinions on Social Media

In order to influence audiences, a number of thematic topics need to exploited. For one, marketeers need to known personal interests of audiences right from their personality traits to income levels. Second, marketeers often use a policy of divide and conquer to successfully conquer markets. This often leads to the rise of echo-chambers and filter bubbles which is often clouded by misinformation. Finally, marketeers use a variety of methods to post content on social media. This ranges from automated scripts(bots) posting content to large teams of human beings often known as "Troll Farms" planning such campaigns. This section tries to contextualise our project along these thematic lines.

3.2.1 User Interests and Privacy

The study of Facebook advertisements has been limited to : a) privacy of users and b) micro-targeting users with their Facebook Interests.

In [34], authors explore Facebook Interests assigned to users and find that many such interests are sensitive in nature. These interests usually belonging to topical categories of religion, politics and sexual orientations. The authors reach to a conclusion that Facebook labels sensitive interests to user profiles. These sensitive interests may further be used by malicious actors, making users on Facebook vulnerable. Such a scenario is depicted in Figure 30 where highly sensitive information can be used to phish users in a geographic location.

Assignment of interests to users on Facebook has always been debatable. It is only recently that Facebook tries to explain to users why they have been assigned certain interests. In many cases, these assigned interests are highly private and sensitive. Irrespective of the fact whether these issues are sensitive or not, security breaches can pose potential threats to users.

3.2.2 Discrimination Of Users

Data from Facebook advertisements can often be used to discriminate between groups of individuals. Discrimination of audiences on a digital platforms is not a new notion. Many platforms try to customise their content for enhanced user experiences. However, the process of enhancing user experiences often leads to discrimination between users.

In [40], the authors study a number of online services such as Netflix, Amazon and Yelp. The findings reveal that some online services discriminate between users on the basis of operating systems, browsers, previous purchase histories and browsing habits. In [47], the authors show that the discrimination is not just limited to prices but also search results.

Discrimination of users on social media platforms is not just limited to various socio-economic groups, user interests or user locations. Users are often discriminated on the basis of their gender. In [41], the authors explore discrimination in marketplaces such as Fiverr [3] and Task Rabbit [4]. The authors find that freelancers on these platforms are often discriminated on the basis of their gender and race. In [39], the authors note that African American passengers on online services such as Uber and Lyft frequently suffer delays in their travel. They attribute the causality of such delays to perceived racial discrimination by drivers when they receive the passenger profile.

[3]https://www.fiverr.com/
[4]https://www.taskrabbit.co.uk/

3.2.3 Target Marketing in a Political Context

Marketeers often discriminate between users based on the basis of their personality traits. In [44], the authors explore a set of advertisements that have been shown to users in the run up of the presidential elections in 2016. They discover a total of eight groups that seem to have published malicious advertisements bordering sensitive topics. The IRA is one of these eight groups. The authors observe that different locations in USA were targeted with different topical issues. They also note that the income level and race of users were used as micro-targeting parameters. In [50], the authors mention that the advertisements published by the IRA used race as a topical issue.

Both of the studies [44] [50], make claims of advertisements being discriminative by race and location. However, neither mention what is their parameter to deduce how an advertisement is discriminative. Further, both studies fail to quantify the effectiveness of these advertisements.

Research in the domain of Facebook advertisements if often limited to analysing data that appears on the feeds of users (crowdsourced data). In this project, we analyse data from the perspective of the publishers.

Further, state of the art literature fails to quantify why advertisement campaigns fails or succeeds. In this project, we try to find a cause for the failure of these campaigns.

3.2.4 Echo Chambers, Filter Bubbles and Misinformation

A divide and conquer strategy by discriminating users based on their personality traits creates a sense of perceived digital spaces. Much like the physical world, individuals of same beliefs and opinion tend to aggregate together. These spaces are often referred to as Filter Bubbles and Echo Chambers. While one may argue that this is formed due to the browsing patterns of individuals, it may also be the case that producers publish content in a fashion which is highly clusterted leading to such bubbles.

In [32], the authors analyse a set of partisan websites that were active during the USA 2016 presidential elections and discover that producers tend to publish as well as refer sites that align to their ideologies and world view.

In [30], the authors explore the phenomenon of echo chambers. The authors observe that there is a polarization of audiences for certain topics. Discussions which are politically oriented seem to be more polarized than discussions of current affairs. The authors go on to say that users receive information of a diverse range of topics on their feeds. However, users choose to interact with topics that match their ideologies.

On the flip side, some researchers strongly believe that the news feed of users are highly curated and polarised [35] leading to an echo chamber like effect. In [32], the authors explore page liking patterns of users on Facebook. They find that users who like a left or right leaning page on Facebook, hardly tend to like a page of the opposing view. Such clustered behaviour may often lead to a myopic vision. Besides, malicious actors may populate these clusters with discriminative content that can affect the ideologies of individuals. In [38], the authors propose a methodology to connect such echo chambers to make users aware of contrasting ideologies as an attempt to reduce polarization in society.

In [55], the authors explore the Twitter ecosystem for mass shooting events in USA. They observe clustered thematic schemes of conversations in groups having ulterior motives. This shows us that there tends to be an aggregation of users in clusters on social media belonging to the same set of ideologies.

3.2.5 Twitter Trolls

An effective method to influence opinions of users on the internet is by creating fake accounts and planting them in social circles of users. These accounts start spreading information which an agenda. They interact with profiles of users which have a higher level of trust and try to tap into their following. Such accounts are often known as sockpuppet accounts [46]. It was discovered that in order to influence the USA 2016 elections, the IRA had employed about 90 individuals to create such sockpuppet accounts to diffuse misinformation [22][25] in the Twitter ecosystem.

Researchers have been focussed in identifying automated social media accounts known as bots. These accounts tend to spread misinformation to reach users of certain demographics to influence their opinion [53]. It is believed that bots have been used to influence multiple elections masquerading as political actors [37] [48] [42]. However, the efforts made by the IRA in the US presidential elections is not limited to bots.

In [61], the authors study the difference of behaviours between bots and accounts which are engineered by teams of humans (trolls). They find that trolls bear a very small degree of influence in manipulating the opinion of audiences. They also find that trolls target specific topics and individuals to gain maximum viewership. Trolls are often vulnerable to be detected by their ever changing opinions on social media. Thus, to mask their identity they delete historical tweets. Since such trolls are often state sponsored and tweet from remote locations, they mask their geo location by using virtual private networks to appear authentic.

Previous works shed light on the characteristics of trolls in terms of the content they spread. This depicts a one way direction of communication with users. It is not clear whether users recognise these troll as legitimate accounts and voluntarily communicate with them instead of just viewing or retweeting. In this project, we measure the mention network of trolls on Twitter to analyse their degree of influence.

4 Dataset Specifications

In this section, we provide an overview of the three focal datasets in this project : a) Facebook advertisements, b) Twitter Trolls and c) Google Trends. Finally, we enumerate on the methods used to collect data as well as the cleaning procedure to obtain coherent structured data.

4.1 Dataset Overview

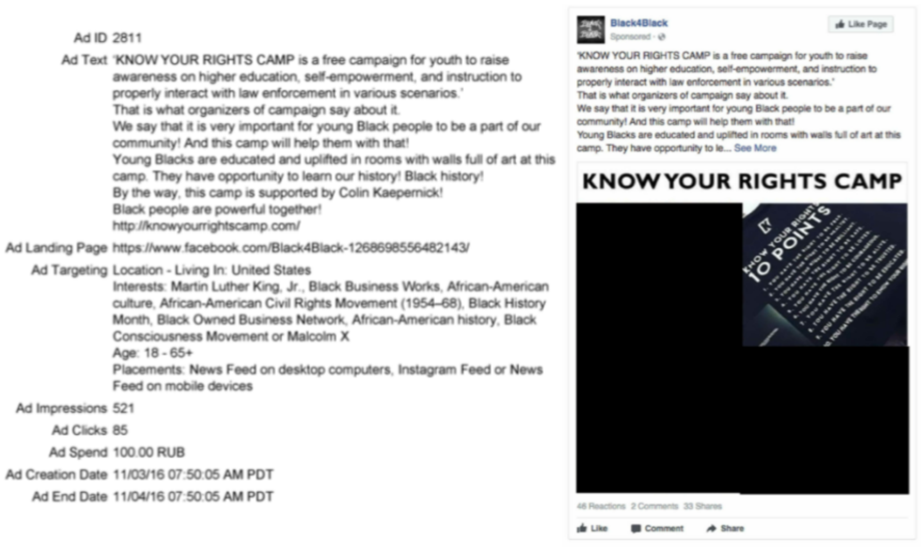
4.1.1 Facebook Advertisements

The U.S House of Representatives Permanent Select Committee notes that were 3,394 advertisements linked to the IRA in between 2015 and 2017. Out of these 3,394 advertisements, 3,517 were made available to the public domain. The advertisements were arranged by quarters of each year ranging from 2015 Quarter 2 - 2017 Quarter 3 (9.02 GB). However, there are no recorded advertisements in June, 2017. In addition, the number of advertisements in July and August in 2017 dwindle to seven and one respectively. These dates correspond to the time when Facebook formally launched an investigation and started banning pages linked to the IRA. We deduce that this is the reason why the third quarter in 2017 have an exceptionally low number of advertisements [13] [29].

To the best of our knowledge, this is the only dataset which throws light on Facebook advertisements published by the IRA. Neither Facebook nor the investigating committee provide the methodology used to discover these advertisements. One may argue that there may be more advertisements published by the IRA. However, this is the closest to the ground truth available in the public domain at the moment.

This dataset is the focal point of our study. It provides us an insight into the targeting practices used by the IRA, the expenditure on each advertisement, the number of individuals who viewed or clicked these advertisements as well as the content and duration of these advertisements. We also get an insight into the pages which published these advertisements as well the landing pages associated with each advertisement.

Each advertisement constitutes of one PDF File. The file describes the advertisement and its corresponding outcome. The parameters which describe the advertisement are listed in Table 1. Figure 5 displays such a PDF file.



Ad ID	2811
Ad Text	'KNOW YOUR RIGHTS CAMP is a free campaign for youth to raise awareness on higher education, self-empowerment, and instruction to properly interact with law enforcement in various scenarios.' That is what organizers of campaign say about it. We say that it is very important for young Black people to be a part of our community! And this camp will help them with that! Young Blacks are educated and uplifted in rooms with walls full of art at this camp. They have opportunity to learn our history! Black history! By the way, this camp is supported by Colin Kaepernick! Black people are powerful together! http://knowyourrightscamp.com/
Ad Landing Page	https://www.facebook.com/Black4Black-1268698556482143/
Ad Targeting	Location - Living In: United States Interests: Martin Luther King, Jr., Black Business Works, African-American culture, African-American Civil Rights Movement (1954–68), Black History Month, Black Owned Business Network, African-American history, Black Consciousness Movement or Malcolm X Age: 18 - 65+ Placements: News Feed on desktop computers, Instagram Feed or News Feed on mobile devices
Ad Impressions	521
Ad Clicks	85
Ad Spend	100.00 RUB
Ad Creation Date	11/03/16 07:50:05 AM PDT
Ad End Date	11/04/16 07:50:05 AM PDT

Black4Black
Sponsored ·

'KNOW YOUR RIGHTS CAMP is a free campaign for youth to raise awareness on higher education, self-empowerment, and instruction to properly interact with law enforcement in various scenarios.'
That is what organizers of campaign say about it.
We say that it is very important for young Black people to be a part of our community! And this camp will help them with that!
Young Blacks are educated and uplifted in rooms with walls full of art at this camp. They have opportunity to le... See More

KNOW YOUR RIGHTS CAMP

[redacted]

Figure 5: Visual Representation of a PDF File describing a Facebook Advertisement in the dataset

4.1.2 Twitter Trolls

The U.S House of Representatives Permanent Select Committee published 3,841 Twitter handles which are believed to be trolls linked to the IRA. The methods used to identify these trolls are not made public. However, this is the closest representation of the truth, in terms of Twitter accounts linked to the IRA currently available.

As a part of the investigation, Twitter has suspended all the accounts which were linked to the IRA [26]. As per Twitter's policy, when an account is suspended, all of its tweets and retweets are made unavailable. One possible method to method to recover such data would be by using web archives such as waybackmachine [5] and the Library of Congress (LOC) [6]. However, waybackmachine does not have a comprehensive listing of these accounts and the LOC currently does not provide access to researchers.

[5]https://archive.org/web/
[6]https://www.loc.gov/

Tweets and retweets provide an excellent description of the network of users a particular account has communicated with. However, it only provides us an insight of a one way communication. In order to understand communication patterns of users who have voluntarily interacted with these troll accounts, we look at Twitter mentions.

Users can mention an account in their tweet by prefixing the Twitter handle of the mentioned account by the @ symbol. Even if an account is suspended, their mentions are still available. One can simply do a string search and get mentions of a particular account.

To this extent, we crawl all mentions of these 3,841 troll handles. This forms our Twitter Troll Mentions dataset which sheds lights on the type of accounts which interacted with these trolls as well as their network topologies.

4.1.3 Google Trends

Google Trends[7] is a platform which provides a trend score for interests over time in a defined geographical region. This score ranges between 0 and 100. A score of 0 depicts no or very low interest in the selected geographic region for the corresponding time range. A score of 100 depicts very high interest in the geographic region for the corresponding time range. The score is also normalised with respect to the date time range selected. The platform also provides related search queries and related topics for every interest.

The data of Google Trends is obtained from an aggregation of google searches by users. The searches cannot be traced back to the users.

In our project, we obtain the Google Trends score of each Facebook Interest corresponding to their targeted location in our dataset. Since Google Trends provides a normalised score, we set our date range from 2015 - 2017 (the time range corresponding to our Facebook Advertisement dataset).

This dataset provides us an insight into real world trends for every interest in our Facebook Advertisement dataset. We will use this dataset to evaluate the relevance of micro-targeted Facebook advertisements published by the IRA.

[7]https://trends.google.com/trends/

4.2 Data Collection and Cleaning

This Section describes the process of data collection and cleaning of the three datasets. We use Python 3.6 along with standard libraries such as numpy, pandas and os to accomplish these tasks. We also use a MySQL database to store our dataset after all preprocessings. A high level view of the database schemas are presented in Section A.3.

4.2.1 Facebook Advertisements

The Facebook Advertising dataset consists of 3,517 PDF files. In order to access data from these files, we convert the files into text files and tokenize the data for easy access. Finally, we clean the data to form coherent sense in terms of character encoding and data types.

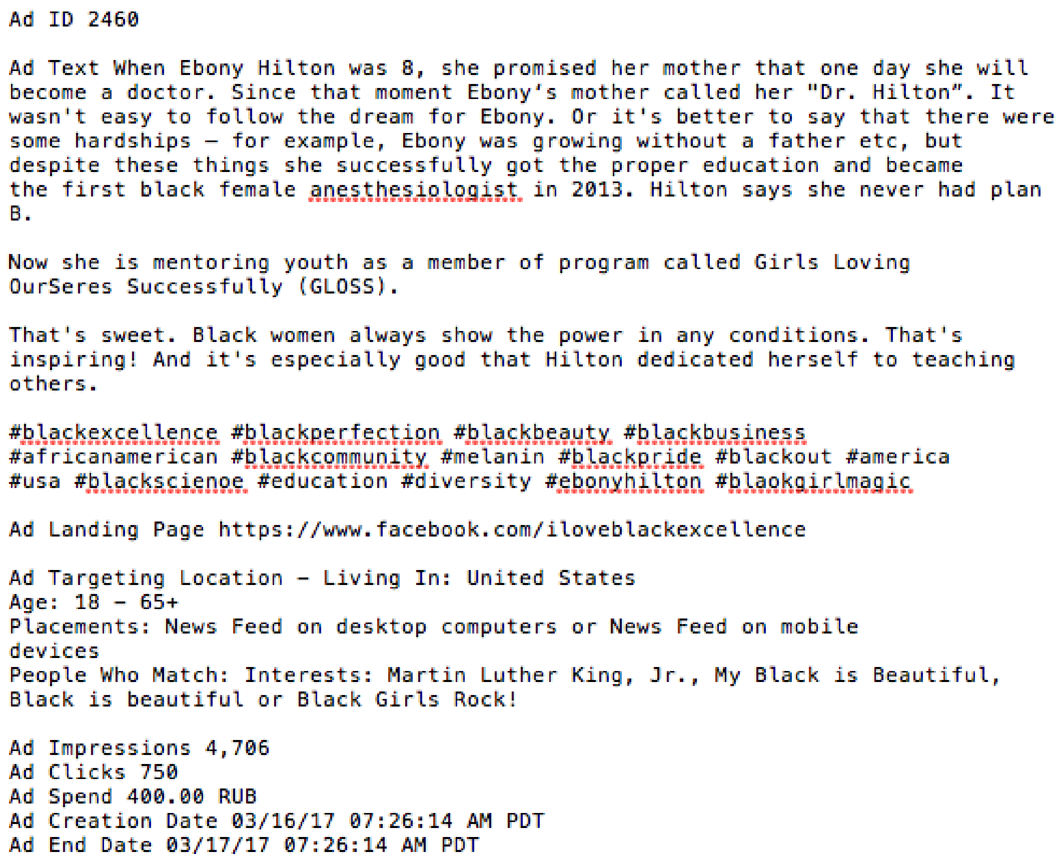
Conversion of PDF to Text Files. To convert all the PDF Files, we use two online services pdf2text.com[8] and pdf2go.com[9]. Initially we use pdf2text.com but this service is rate limited to only twenty files in one batch. pdf2go.com has a rate limit of sixty files in one batch. Using a combination of both these services, we converted all our PDF files to text files.

Tokenization of Key-Pair Values. The text files now contain all the data of the PDF files. A screenshot of the text file corresponding to the PDF in Figure 5 is shown in Figure 6. However, it is not computationally efficient to iterate the files every time to access data during our analysis. We preprocess these files to obtain key-pair values and store them in a SQL database.

We manually go through every file in the second quarter of 2015 to discover the parameters that define a singular advertisement. We further reverse engineer Facebook Ads manager to obtain all possible parameters that can define an advertisement. These parameters are described in Table 1.

We observe some parameters are constant for all advertisements such as Ad Id, Ad Text, Ad Landing Page, Ad Targeting Location, Age, Placements, Ad Impressions, Ad

[8]https://pdftotext.com/
[9]https://www.pdf2go.com/



```
Ad ID 2460

Ad Text When Ebony Hilton was 8, she promised her mother that one day she will
become a doctor. Since that moment Ebony's mother called her "Dr. Hilton". It
wasn't easy to follow the dream for Ebony. Or it's better to say that there were
some hardships — for example, Ebony was growing without a father etc, but
despite these things she successfully got the proper education and became
the first black female anesthesiologist in 2013. Hilton says she never had plan
B.

Now she is mentoring youth as a member of program called Girls Loving
OurSeres Successfully (GLOSS).

That's sweet. Black women always show the power in any conditions. That's
inspiring! And it's especially good that Hilton dedicated herself to teaching
others.

#blackexcellence #blackperfection #blackbeauty #blackbusiness
#africanamerican #blackcommunity #melanin #blackpride #blackout #america
#usa #blackscienoe #education #diversity #ebonyhilton #blaokgirlmagic

Ad Landing Page https://www.facebook.com/iloveblackexcellence

Ad Targeting Location - Living In: United States
Age: 18 - 65+
Placements: News Feed on desktop computers or News Feed on mobile
devices
People Who Match: Interests: Martin Luther King, Jr., My Black is Beautiful,
Black is beautiful or Black Girls Rock!

Ad Impressions 4,706
Ad Clicks 750
Ad Spend 400.00 RUB
Ad Creation Date 03/16/17 07:26:14 AM PDT
Ad End Date 03/17/17 07:26:14 AM PDT
```

Figure 6: Text File corresponding to the PDF in Figure 5

Clicks, Ad Spend and Ad Creation Date. In some advertisements Ad End Date is not present. However, People Who Match is often varying. In most cases People Who Match only consists of Interests. In some advertisements People Who Match also consist of Friends of Connections, Excluded Connections and Language.

Our goal is to extract data from the text file in a tokenized format. We search for the starting index of every parameter in the text file and keep a track of the indices. We know that these parameters occur sequentially in all our files. For example, AD Text is always followed by Ad Landing Page followed by Ad Targeting Location and so on. Once we have the indices of every parameter we simple use index manipulation in our text file to extract the data. The corresponding Python code is presented in Section A.5.

For example, let us assume that the starting index of Ad Text in Figure 6 is x and the starting index of Ad Landing Page is y. We know that the length of the string “Ad Text” is 7. To extract Ad Text we simple extract characters from position x+7 and y in the text file. We further process this extracted string to eliminate leading and trailing whitespaces, resolve character encodings and convert the string to relevant data types.

Data Cleaning. Due to conversion of PDF Files to text files, some portions of text were encoded incorrectly. In addition, the format of Facebook Interests and targeting locations are presented in the natural English language. We need to convert them to simple tokenized keywords. We also preprocess Ad Spend to a float value and extract the currency. The Start and End Time timestamps are also split into the date and time and the timezone is extracted. Finally, we convert all numeric values in string format such as Impressions and clicks to relevant data types. The Python scripts which were used to execute these tasks are listed in Section A.5.

- **Text Encoding** - During the conversion of PDF files, some portions of text, specifically URLs and some interests were encoded in a different character set. We search for characters which were encoded differently and convert them to the UTF-8 encoding.

- **Tokenization of Facebook Interests and Location** - Facebook Interests and Targeting Location are presented in the natural English language. We extract the relevant information and store them in our database.

- **Tokenization of Ad Spend, Ad Creation Time and End Time** - Ad Spend has two attributes the spend value and the currency. We extract the relevant data to form tokens. Ad Creation and End Time are timestamps which have date, time and timezone. We extract these three parameters and store them in our database.

- **Preprocessing of Impressions, Clicks** - Impressions and Clicks are essentially integers. However, they are present as strings in our dataset. We convert the strings to integer data type and store it in our database.

The summary of the dataset is presented in Table 2.

Parameter	**Description**
Ad Id	Generic Integer Value. This parameter is added by the investigating committee publishing these advertisements. They do not have any relevance in our study or in the description of Facebook Ads
Ad Text	The textual description accompanying the advertisement. This piece of text also carries relevant hashtags
Ad Landing Page	The URL of the destination page when the advertisement is clicked
Ad Targeting Location	Describes the geographical region that is being targeted
Age	Age Group of demographic being targeted
Placements	Describes the placement locations of the advertisement
People Who Match	People who match consists of a number of parameters such as Interests, Friends of Connections and Excluded Connections
Interests	Describes the Facebook Interests of users that are being targeted
Friends Of Connections	Describes other pages whose users and friends of users are being targeted. The publisher of the advertisement must own the other pages whose connections are being exploited
Excluded Connections	Advertisers can exclude a set of followers of particular pages they own
Language	The Language being targeted
Ad Impressions	Defines the impressions of the advertisement. In simpler words, the number of times the advertisement is viewed
Ad Clicks	The number of times the advertisement is clicked
Ad Spend	The total expenditure for the advertisement
Ad Creation Date	The date timestamp of when the advertisement was created
Ad End Date	The date timestamp when the advertisement became inactive

Table 1: Description of parameters that define a Facebook Advertisement

Total No. Of advertisements	3,517
Total Impressions	4,05,72,728
Total Clicks	37,34,779
Total Ad Spend	5856737.1 RUB

Table 2: High Level Summary of the Facebook Advertising Dataset

4.2.2 Twitter Trolls

The Twitter Troll dataset consists of 3,841 Twitter Troll handles. Our goal is to obtain all mentions of these handles. Further, we classify each account which has mentioned these trolls as : a) Twitter Verified Accounts, b) Bots and c) Random Twitter Users. A summary of the Twitter Mentions dataset is presented in Table 3.

Crawling Twitter for Mentions. We formulate a Twitter mentions dataset by crawling all the mentions for every troll account. In order to crawl Twitter we use a Python library known as Twitter Past Crawler[10]. The library queries for tweets which mention our trolls on Twitter. The query returns a set of tweets where the troll handle was mentioned. In addition, the query also return the tweet id (unique id identifying the tweet), account name (name of user which is different from the handle), user id (unique numeric id locating an account on Twitter), timestamp of tweet, text of tweet, links attached with the tweet, retweet count, like count and favorites count. The Python script to crawl all mentions is listed in Section A.5. It takes a total of seventeen hours to crawl all the mentions.

Cleaning the Fetched Tweets. The Twitter mentions dataset needs to be cleaned further as some tweets are malformed due to error during crawling. We eliminate these tweets. The remaining tweets are stored in a SQL database in a tokenized format.

Identifying Twitter Verified Accounts. We query Twitter with each user id using the R Twitter Package [11]. The query returns whether a user id is a verified Twitter profile or not. In addition, it also returns a number of other details about the user id such as screen name, location, following count, follower count, number of tweets, account creation date, language, profile url and a boolean variable which specifies if the account is protected or not. The R Script to query information about a user id on Twitter is listed in Section A.5.

Identifying Bots. Finally, we want to identify users which are bots who have

[10]https://github.com/keitakurita/twitter-past-crawler

[11]https://rtweet.info. It must be noted that the script used to query user details by user id was written by Pushkal Agarwal, a PhD student in the Department of Informatics

mentioned our trolls. We use Botometer [12] a popular library which returns a number between 0 and 1 as a Bot Score for a Twitter handle. A score of 0 means that it is highly unlikely that the account is a bot. A score of 1 means that the account is in all probability a bot. This library samples a set of tweets for every user id and generates the bot scores over this set of tweets.

Our dataset consists of 1,91,883 unique twitter users as mentioned in Table 3. It is computationally not feasible to sample a huge number of tweets for each user and then pass it through the Botometer to get a bot score. In addition, if a user is a bot, it is highly likely that they will tweet frequently as that is the primary objective of a bot. To this extent, we plot a CDF of the frequency of tweets of each unique user. The CDF is presented in Figure 7.

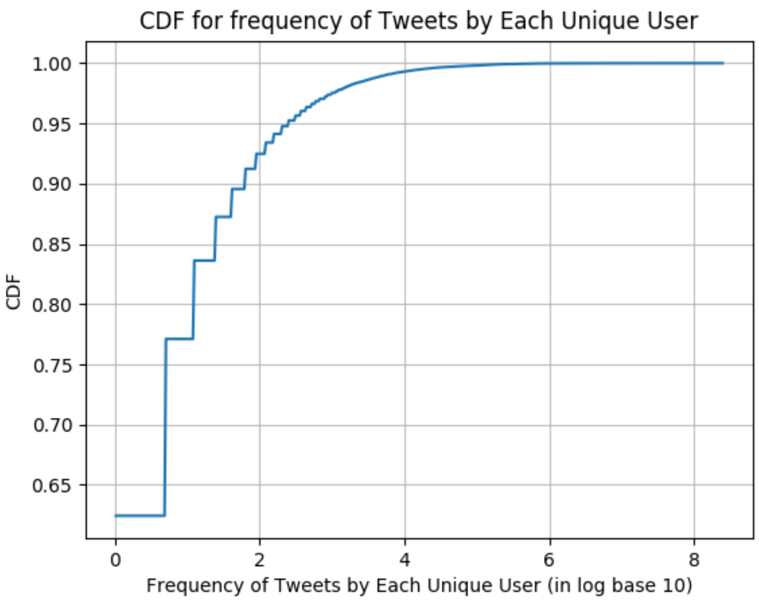


Figure 7: CDF of Frequency of Tweets by each Unique User

From Figure 7, we note that if a user is picked at random, there is a chance of 80% that it tweets less than 10 times (Note, the X-Axis is in log base 10. log base 10 (10) = 1). Further, we note that 11,268 users tweet at least 10 times. These 11,268 users contribute

[12]https://github.com/IUNetSci/botometer-python

No. Of Troll Accounts	3,841
No. Of Troll Accounts who have Twitter Mentions	2,930
Total Tweets Collected	7,06,957
Total Tweets Preserved (out of the collected Tweets)	6,82,819
Total No. of Unique Users	1,91,883
Total No. Of Twitter Verified Profiles	1,058
Total No. Of Bots	530

Table 3: High Level Summary of the Twitter Mentions Dataset

to 55.08% of the entire Twitter mentions dataset. We safely select a cut of 10 and run the Botometer on accounts who have tweeted at least 10 times. The corresponding code to generate the CDF and fetch bot scores is presented in Section A.5 [13]. We classify an account to be a bot if the bot score is greater than or equal to 0.5. We base this threshold based on previous works [52] [59] [31] [58].

4.2.3 Google Trends

Google does not provide a Python API to query Google Trends. We use a popular Python library known as Pytrends[14] to fetch Google Trends data. Since the Google Trends is rate limited, we fetch our data over a span of two days. The relevant Python scripts are listed in Section A.5

Google Trends for entire USA. We first collect trend scores for every Facebook Interest in our Facebook Advertisement dataset for the entire USA geographic region. We save the data in the form of CSV files. Each file contains 157 scores representing 157 sampled dates over our time range (2015 - 2017).

Google Trends for every State. Finally, we want to collect the trend scores of every Facebook Interest with respect to every state in USA. We note that all Facebook Interests are not used for targeting in every state.

Each (interest, state) query returns a list of 157 sampled trend scores over our date time range. We save the list in a CSV file. Finally, we find the average of the 157 sampled

[13]It must be noted that the script to fetch Botometer scores was written by Pushkal Agarwal, a PhD student in the Department of Informatics

[14]https://github.com/GeneralMills/pytrends

trend scores and save it in a SQL database.

5 Methodology and Implementation

In this Section, we provide an overview of theories and methods that are used in this project. We briefly describe the background behind each theory and provide an overview of its corresponding implementation in this project.

5.1 Advertisement Evaluation Metrics and The AIDA Model

The domain of advertisements in the digital sphere is often evaluated by a standard set of metrics. These metrics are defined as follows :

- **Impressions.** The total number of views received by an advertisement. In the domain of Facebook advertisements, this includes duplicate views by the same user.
- **Reach.** The total number of unique users who have viewed the advertisement.
- **Clicks.** The total number of clicks received by the advertisement.
- **Ad Spend.** The total amount of money spent on the advertisement. The spend maybe lower than the budget set for the advertisement. It may be the case that the advertisement runs for the entire duration but does not exhaust the budget.
- **Click Through Rate (CTR).** The ratio of clicks to impressions.
- **Cost Per Mile (CPM).** The cost of an advertisement per 1,000 views. (Total Spend / (Total Impressions/1000))
- **Cost Per Click (CPC).** The cost of an advertisement for a singular click. (Total Spend/Total Clicks)

In this project, we compare the benchmark data of these metrics across all industries in USA with our dataset. It provides us with an intuition of the performance of the advertisements as well as an understanding of the campaign strategies.

Marketeers often refer to the **AIDA model** to design a high performance advertisement. This model was introduced by Elmo Lewis in 1906 and popularized by Edward K.

Strong, Jr. [57]. Although this model was published to evaluate physical advertisements, it extends to the digital domain. The model states that there are four clear stages a user goes through before clicking on an advertisement, which is the ultimate goal. The model is pictorially described in Figure 8.

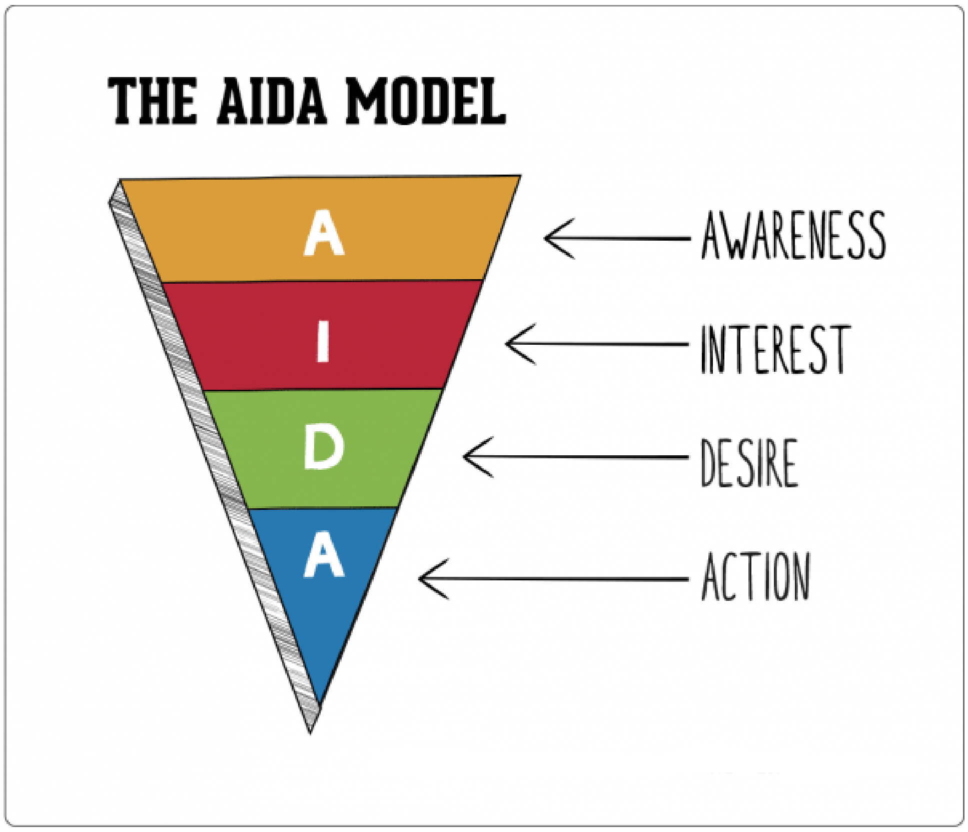


Figure 8: Pictorial Description of the AIDA Model [10]

First, users need to be ***aware*** of the topical background of the advertisement. This stage assumes that a user needs to connect with the thematic background of the advertisement to further spend time on the advertisement space. Second, the advertisement needs to capture the ***interest*** of the user. If the user is interested, then the user engages with the advertisement. Engagement can be in the form of simply reading the advertisement. This stage can be defined as ***desire*** to ultimately click on the advertisement. The stages of interest and desire are highly dependant on the content of the advertisement. Finally, the user may click on the advertisement which corresponds to an ***action*** by the user.

The AIDA model is often known as the “purchase funnel” [51] and used as a standard practice to design advertisements. In this project, we use the AIDA model to draw inferences on the quality of the campaign drawn out by the IRA.

These metrics(CPC,CTR,CPM) and the AIDA model judge the quality of an advertisement in terms of selling a commodity. However, the goal of the IRA is to influence opinions of audiences which is may not be directly proportional to garnering clicks. Impressions can be effective in influencing audiences as well. In this project, The AIDA model along with these set of standard metrics, at best, can provide us with an intuition of the quality and targeting practices of the IRA campaign.

5.2 Micro-Targeting Parameters As Conditional Probability

Simmons OneView is an established market research methodology used to analyse market demographics [54]. In [44], the authors expand on this index to build a probabilistic model that throws light on the targeting parameters used by suspicious political campaigns. The index provided in [44] is presented in Equation 5.1.

$$100 \times \frac{P(X=1|Y=1)}{P(X=1)} \tag{5.1}$$

Equation 5.1 provides a probabilistic measure of how likely it is that a particular geographical region will be targeted with a certain topical issue. For example, we may be interested to know if Kansas is a state where advertisements related to LGBT issues are likely to be published. We would thus measure P(Kanas|LGBT). This would be equal to the number of LGBT advertisements published in Kansas over the total number of LGBT advertisements published across the entire country. A value of P(Kansas|LGBT) tending to unity would mean that LGBT topics were exclusively used as a targeting parameter in Kansas.

In our project, we refine Equation 5.1 and present it as Equation 5.2. We use this conditional probability to discover the micro-targeting parameters used by the IRA to reach audiences.

$$P(State|Interest) = \frac{\text{No. Of advertisements in State X with Interest Y}}{\text{Total No. of advertisements with Interest Y}} \tag{5.2}$$

5.3 The Swing Vote

The USA Presidential Election is based on the Electoral College. The Democrats and the Republicans nominate one member each to represent their parties. Voters ultimately vote either Republican or Democratic. In a scenario where a voter is not happy with the representation from either parties, they are left undecided with their vote. These votes are known as “swing votes”. The “swing vote” was considered to be a deciding factor in the 2016 elections [6].

Due to the Electoral College based voting system, certain states have consistently voted either democratic or republic since 1996. These states are considered to be safe zones for either parties. The election is said to be actually battled in “Swing States”. Swing states are states which have voted either democratic or republican since 1996 without any consistent patterns. It is often noted that election campaigns put maximum efforts in these states.

In this project, we want to quantify if the Audience Reached by the IRA published advertisements was significant enough to have influenced the election. In order to do so, we first calculate the Audience Reached by the IRA in every state using Equation 5.3. This provides us with a measure of the percentage of population that has either viewed or intereacted with the IRA published advertisements.

$$\text{Audience Reached} = 100 \times \frac{\text{Total Impressions/Clicks in State X}}{\text{Population of State X}} \tag{5.3}$$

Finally, we want to understand if the percentage of population which was reached by the IRA was significant enough to have influenced the election. We use the number of undecided votes or the “swing vote” to act as a benchmark to understand the degree of influence resulting from the IRA campaign.

5.4 Related Facebook Interests

Facebook Interests are used in the advertisements laid out by the IRA to reach audiences. These interests can be searched for in the Facebook Ads Manager using a text field. If the interest that the user is searching for is a match in the Facebook Interests database, they can choose it as a targeting parameter. This means that an interest belonging to a certain topic may correspond to multiple Facebook Interests.

For example, let us locate advertisements that are topically related to LGBT issues in our dataset. We find that our dataset has multiple Facebook Interests that cover the spectrum of LGBT issues. These interests are LGBT equality, LGBT community, LGBT culture, LGBT history, LGBT right by country, LGBT social movement, Rainbow Flag (LGBT Movement) and Same sex marriage. There are 8 interests that can fall under the same topical issue.

LGBT issues is a straightforward example which clearly shows that 8 unique interests cover the same topical issue. In a spectrum of 775 unique interests, multiple issues maybe related to each other which is not explicitly visible to the naked eye. In order to group these interests under their respective topical umbrellas, we use hierarchical clustering.

There are multiple methods to group data. One of the main reasons we use clustering is because our dataset is not labelled. In addition, we want to view the degree of association between each interest in our dataset. Hierarchical clustering helps us to visualise the correlationship of each Facebook interest against a scale.

In this project, we use hierarchical clustering to view the correlationship of Facebook interests with one another. Further, we define a cutoff threshold to form topical groups. Each topical group would span a spectrum of similar interests. We use these topical groups to analyse the topical relevance of advertisements across targeted geographical locations.

5.5 Campaign Objectivity

A successful advertising campaign usually has a clear topical objectivity. In this project, we want to evaluate if the campaigns drawn out by the IRA had significant objectivity on topical issues.

In [33], the authors use two metrics **Focus** and **Entropy** to measure the popularity of YouTube videos in a local geographical region compared to a global region. Focus and Entropy are defined in Equations 5.4 and 5.5. We extend these metrics to our project.

$$\text{Focus of State X} = \frac{1}{V_i} \times \max\,(\text{Interest Y}_i) \tag{5.4}$$

$$\text{Entropy of State X} = -\sum_{k=1}^{N} \frac{\max\,(\text{Interest Y}_k)}{V_i} \log_2 \frac{\max\,(\text{Interest Y}_k)}{V_i} \tag{5.5}$$

Note, V_i refers to the total number of advertisements in State X.

Equation 5.4 gives us a measure of the focus of an advertisement campaign in a particular geographical region. The focus is a ratio of the number of advertisements in a topical group appearing the maximum number of times in a geographical region to the total number of advertisements in that geographical region. For example, if Kanas has 12 advertisements in total, with 10 advertisements whose topical issues are aligned to LGBT issues and the other two advertisements have their topical issue as Patriotism, then the Focus of Kansas is 0.83. This is obtained by dividing the number of advertisements that belong to the topical category which appears the most number of times (in this case LGBT with 10 advertisements) over the total number of advertisements in Kansas (in this case 12). If the value of Focus is unity, it means that the campaign was focussed with a singular objective.

Entropy on the other hand provides a measure of how random the advertisement campaign was over the entire geographical region. A high value of entropy suggests a randomly organised campaign which would tend to have very little effect on audiences.

In this project, we use the notion of focus to measure how focussed the advertisements campaigns were in each state. We can calibrate the results against every Facebook Interest (Unclustered Interests) and the topical groups (Clustered Interests) we obtain from the grouping of Facebook interests as per Section 5.4.

The notion of Focus will help us to determine why the campaigns drawn out by the IRA did not have a substantial degree of influence. A well planned campaign ideally should be very focussed with a clear topical objectivity.

5.6 The Truth Serum

A truth serum is a form of drug which enforces individuals to yield the truth. Authors in [56], propose a digital version of the truth serum. The authors claim that users on social media platforms tend to socially curate themselves in order to preserve a socially acceptable image. However, individuals tend to portray an honest version of themselves under the veil of anonymity.

Google Trends aggregates searches made by users. Users cannot be mapped to these searches and their anonymity is preserved. This in turn enables users to reflect a true version of themselves without any sort of curation. The authors use this notion to predict polling trends across various states in USA by tracking for searches such as “how to vote” and “where to vote”. They note the correlation between the actual poll trends and the Google trends searches are very high. The traditional method to predict polling trends is by using survey forms (which do not preserve anonymity). The authors further note that the actual polling trends and data from the survey forms do not correlate.

The authors extend this concept to locate geographical regions in USA which are racially sensitive. Racial searches were tracked across the country and the authors concluded that the eastern half of USA tends to be more racial sensitive than the western half.

In this project, we use the “Digital Truth Serum” to analyse if the topical issues which were used by the IRA to target audiences were relevant geographically. If the IRA intended to influence audiences, the advertisement campaigns would need to echo the real topical interests of these geographic locations. To this extent, we draw a comparison between the topical interests used by the IRA and the actual regional topical interests. We obtain a ranking of the regional topical interests using Google Trends. Finally, we draw a correlationship between the actual topical interests and the interests in our dataset. This correlationship would provide us with a measure of the topical relevance of the advertisements served by the IRA.

6 Results, Analysis and Evaluation

6.1 Generalised Overview of IRA Published Facebook Advertisements

The Facebook Advertisements dataset consists of 3,517 advertisements. To throw further light into the campaigning practices of the IRA, we investigate a few metrics which were introduced in Section 5.1.

Facebook advertisements published by the IRA were viewed 4,05,72,728 times (total impressions in our dataset) and clicked on 37,34,779 times (total clicks in our dataset). Out of these 3,517 advertisements, 828 advertisements were targeted to specific states in US and 90 advertisements were targeted to other countries than the US.

According to US Census data, the population of USA in 2016 was 32,34,05,935 [27]. We want to quantify the percentage of US population which viewed or interacted with these advertisements. We find the total percentage of US population that was exposed to these advertisements by aggregating the percentages of US population exposed to each advertisement. We find that 0.2884% of the US Population has viewed these advertisements and 0.0159% has interacted (clicked) with them.

The IRA spent 5,84,990 RUB for 3,512 advertisements and 109 USD for 5 advertisements. If we convert the total amount to RUB at a conversion rate of 61.90, the total expenditure amounts to 58,56,737.10 RUB. This is equivalent to 94,616.10 USD. According to Glassdoor[15], the average salary of a CIA[16] Intelligence Analyst in Washington is 97,917 USD per annum [7]. In 2016, Evgeny Buryakov, a Russian spy revealed his salary to be approximately 204,000 USD per annum [19]. This shows that a full fledged campaign to influence audiences can be launched digitally with a salary less than that of singular intelligence agent.

In order to quantify the impression, clicks and spend against a benchmark, we obtain the average benchmark data for CPC, CPM and CTR across all industries in USA [12][11][8]. The values obtained are listed Table 4 and compared with the values in our dataset.

[15]https://www.glassdoor.co.uk/index.htm
[16]Central Intelligence Agency, USA

Metric	Facebook Advertisements Dataset	Benchmark across USA
CPC	0.14 USD	1.72 USD
CTR	7%	0.9%
CPM	7.34 USD	7.19 USD

Table 4: Generalised Overview of the Facebook Advertisements Dataset

The average CPC and CTR in our dataset beat the baseline averages. This might indicate that the performance of the advertisements excelled. However, this is not the case because a select set of advertisements performed exceedingly well while the remaining did not. This variation of performance does not make it feasible to draw conclusions by just comparing the CPC and CTR with baseline values.

If we were to draw further conclusions from the AIDA model based on the number of clicks, we can define a metric **clicks per impression**. Clicks per Impression (CPI) is the total number of clicks obtained by an advertisement over the total number of impressions. It gives us an insight on the quality of the overall campaign assuming the ultimate goal is to get a user to click on an advertisement. The plot so obtained with respect to each quarter in every year is presented in Figure 9 . The general trend of the plot is in a positive direction. However, the peaks are followed by troughes making it difficult to draw a conclusive inference.

We further define a ratio **CPC/CTR**. A general decrease in the ratio would indicate that efficiency of the campaign to receive clicks increases over time, resulting in an overall efficient campaign. The plot is presented in Figure 10. We note that our results are quite the opposite. The ratio keeps increasing upto the first quarter of 2017. It only decreases after the first quarter of 2017 and subsides in the last quarter. Since the elections were held in November, 2016 we can draw upon an intuition that the influence of the campaign was not substantial.

Our dataset does not throw light on *how the ad spend was computed.* The options provided by Facebook to charge an advertisement are with respect to link clicks, landing page views, unique impressions (reach) and impressions. One of the main limitations of the AIDA model is that it can only evaluate an advertisement with respect to the

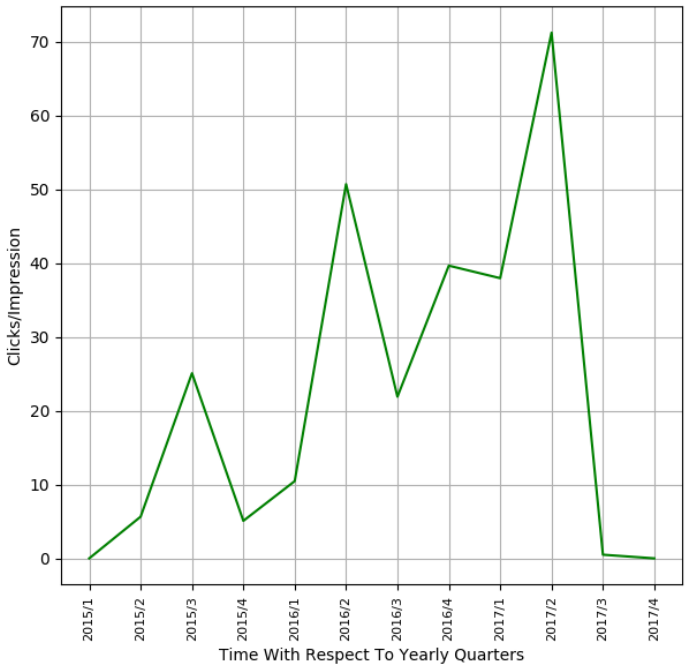


Figure 9: Clicks Per Impression with Respect to each Quarter from 2015-2017

number of clicks it receives. However, it might be the case that the ultimate goal of the IRA was to receives views in terms of impressions and not clicks. We believe this might be the case as the IRA is trying to manipulate opinions via advertisements and not sell a commodity which is usually the norm. This is one of the main reasons why the "purchase funnel" can only provide us with an intuition and not decisive conclusion. If we compare the benchmark data value of CPM across USA with that in our dataset (Table 4), we note they are in alignment. This bolsters our assumption that the IRA was interested in gathering impressions rather than clicks.

The *distribution of advertisements over time* in terms of each quarter from 2015 - 2017 is presented in Figure 11. We can see that the campaign was most active during the election period (both before and after). However, the activity does not recede after the elections as was the case for partisan sites run from Macedonia [32]. Experts believe that the goal of the IRA was much bigger than the 2016 elections, they wanted to cause

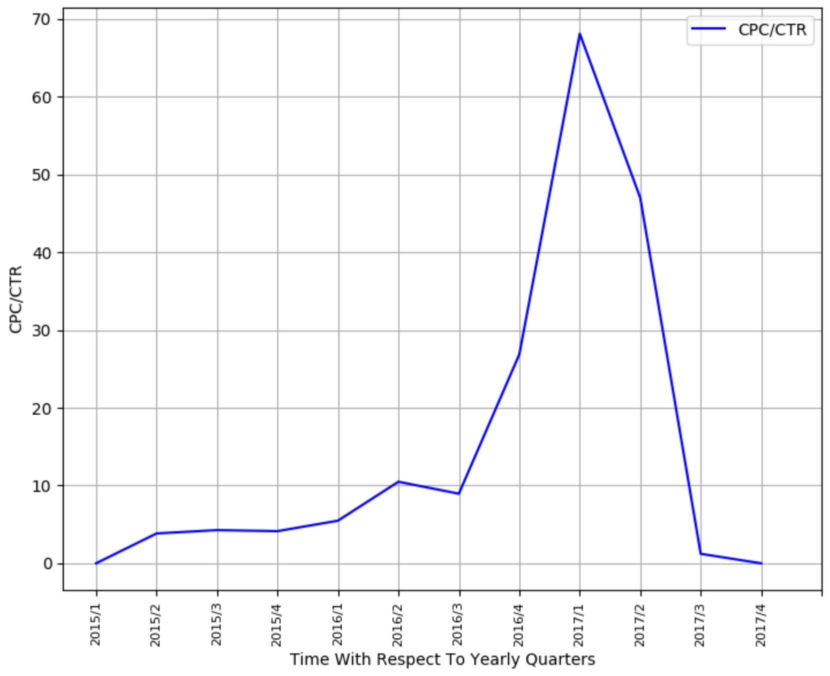


Figure 10: CPC/CTR with Respect to each Quater from 2015-2017

serious damage to the fabric of the society [18].

Out of the 3,517 advertisements, 828 advertisements were targeted directly to states in USA. The frequency of advertisements targeted to specific USA states are presented in Figure 12. (Only states with at least 10 advertisements are presented in Figure 12)

90 advertisements were targeted to countries other than the United States. The countries which were targeted apart from USA are United Kingdom (59), Mexico (13), France (12), Germany (12), Russia (6), Canada (5), Netherlands (4), China (3) and Turkmenistan(1).

Literature claims that users were targeted based upon racial, gun and police issues by the IRA. To build an intuition over this claim, we generate word clouds of the Ad Text and Facebook Interests used to target users (Figures 13,14). It is visible that our dataset illuminates the same nature of targeting as mentioned in literature.

From the overview of the dataset we build an intuition that the IRA campaign was

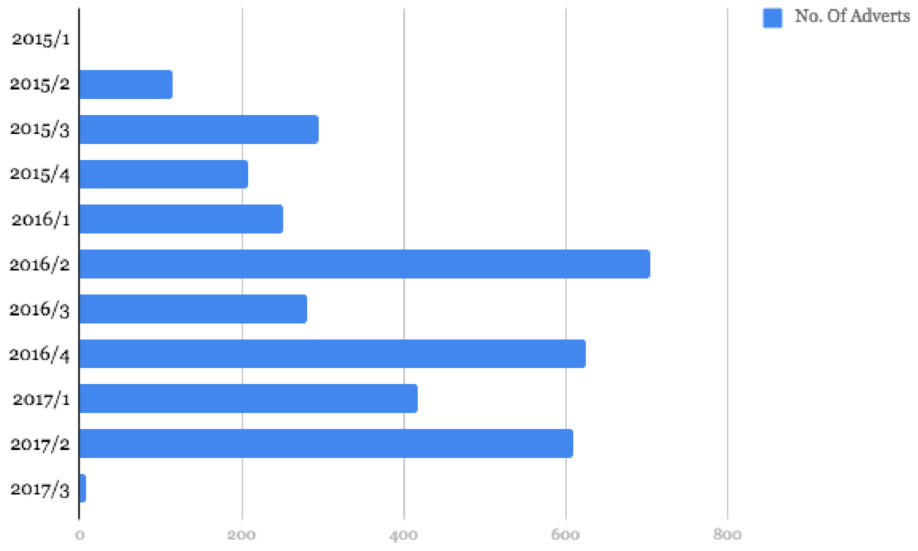


Figure 11: Distribution of Advertisements over Time with Respect to Yearly Quarters

targeted based on location and personal interests of users to garner impressions. We also build an intuition that the overall performance of the campaign was not overwhelming.

6.2 Micro Targeting Parameters in the IRA Campaign

In Section 6.1, we observe that sensitive topical issues were used to publish advertisements by the IRA. We also note that the focus of advertisements were spread across a number of states in USA.

Equation 5.2 is used to generate a probabilistic measure P(State|Interest). P(State|Interest) provides us a measure of how likely it is that an interest Y was used to target advertisements in state X. If P(State|Interest) is equal to or tends to unity, it means that interest Y was exclusively used to target state X. A value tending to zero would mean that interest Y was used as a targeting parameter across a number of states.

We generate P(State|Interest) for every state and Facebook Interest pair in our dataset. We obtain 1,043 such pairs. We generate a Probability Distribution Function(PDF) of these 1,043 pairs as represented in Figure 15.

From Figure 15, we observe that the likelihood that P(State|Interest) to be in be-

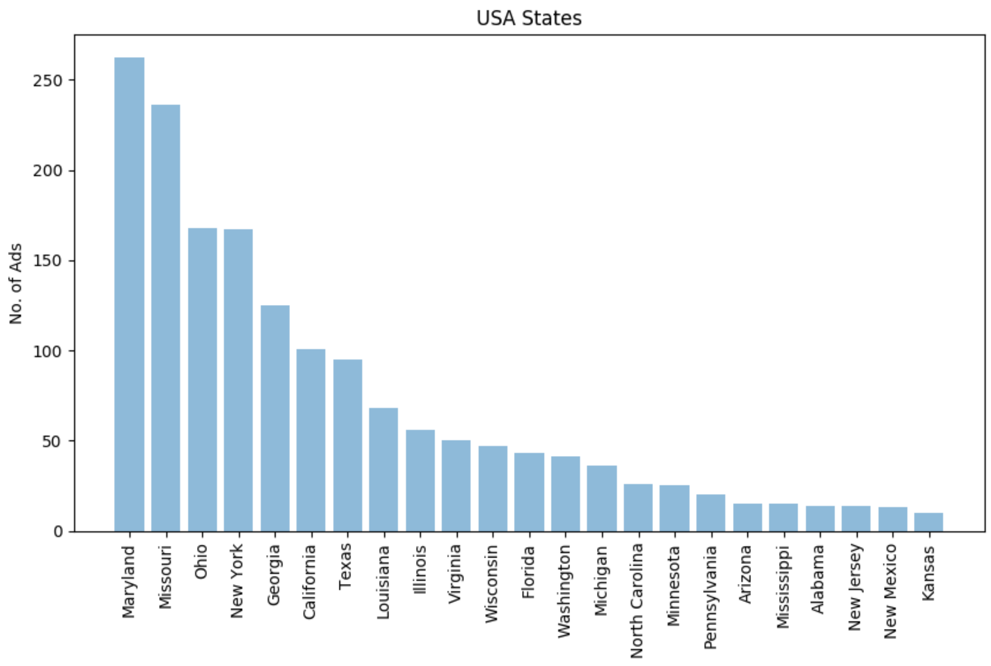


Figure 12: Frequency of Advertisements across the States of USA

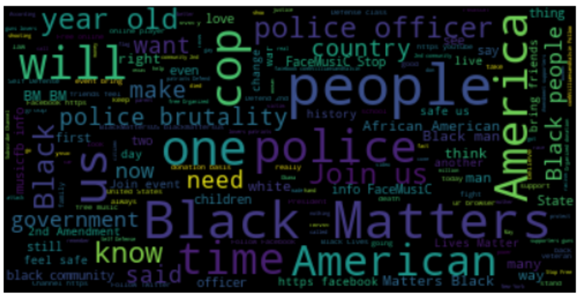


Figure 13: Word Cloud of Ad Text

tween 0.9 - 1.0 is 0.2598. Alternatively, this means that 271 values of P(State|Interest) out of 1,043 pairs are in between 0.9 - 1.0. We also observe, that the likelihood of P(State|Interest) to be in the range 0.0 - 0.1 is 0.3404. This means that 355 pairs of P(State|Interest) out of the 1,043 pairs measure in between 0.0 - 0.1.

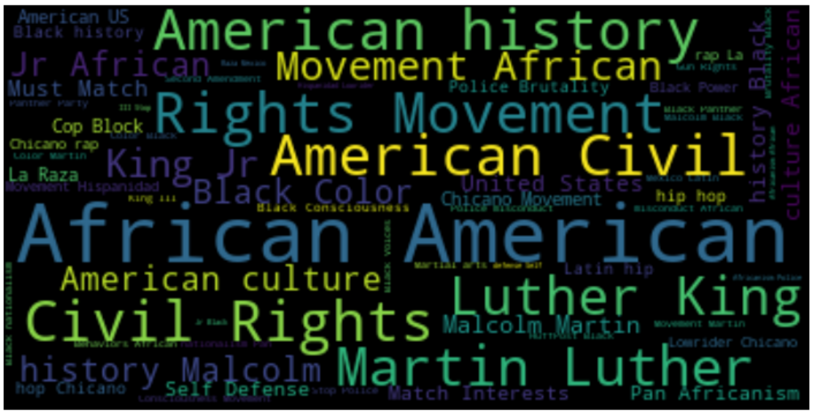


Figure 14: Word Cloud of Facebook Interests

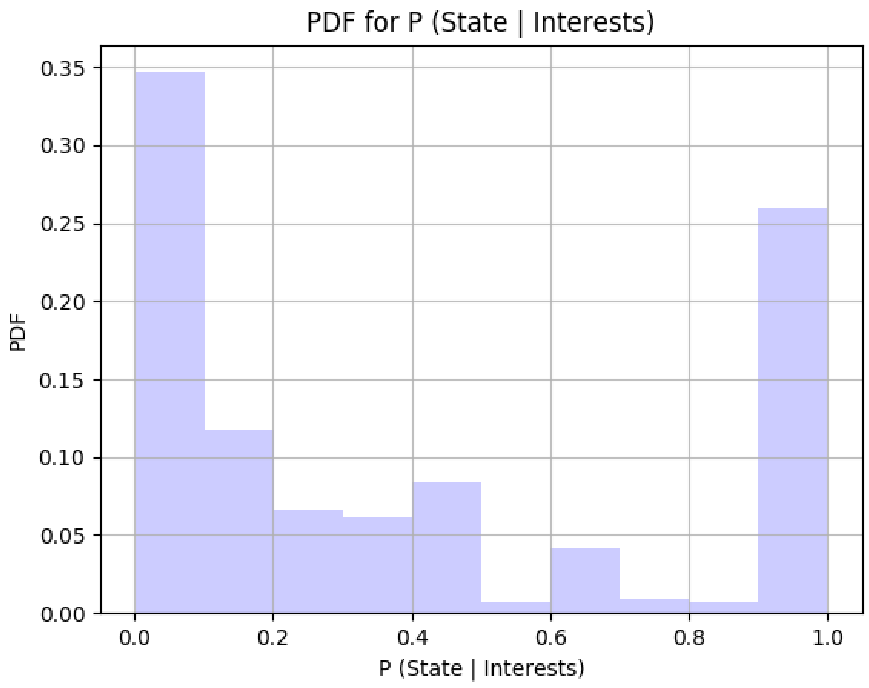


Figure 15: PDF of P(State|Interest) as per Equation 5.2

We note that the PDF is clustered at two ends of the spectrum. This bimodal distribution has two significances. First, the peak of the PDF for values of P(State|Interest) in between 0.0 - 0.1 suggests that a spectrum geographic regions were not targeted with the same interest. The probability of an interest Y being used as a targeting parameter in multiple states is very low. Hence, Facebook Interests alone could not have been used as a micro-targeting parameter to reach audiences.

Finally, the peak of the PDF for values of P(State|Interest) in between 0.9 - 1.0 suggest that a number of interests were exclusively used in certain states. This means that a combination of the geographic location of the user and their personal interests were used as micro-targeting parameters to reach audiences.

We conclude that the IRA led Facebook advertisements campaign tried to exploit regional topical issues to influence opinions of individuals by micro-targeting users based on their location and personal interests.

6.3 Degree of Influence

We want quantify the number of individuals who have interacted with the advertisements published by the IRA in form of impressions or clicks with respect to the US population. In order to do so, we use Equation 5.3. Audience Reached gives us a measure of the percentage of the population of a particular state[17] that has been exposed to these advertisements. Audience Reached can be calibrated in terms of impressions as well as clicks.

Next, we look at the percentage of undecided voters across USA. As discussed in Section 5.3, the Swing Vote plays an important role in USA elections. If the campaign drawn out by the IRA had an influence to any degree, it must have at least influenced the undecided voters.

We calculate the Audience Reached in every state and plot it as a CDF in terms of impressions and clicks (Figures 16 and 17). We note that maximum audience reached in any state in terms of impressions amounts to a maximum of 1.6% of the state population. The maximum audience reached in terms of clicks in any state amounts to a maximum 0.07% of the state population. We can further observe that more than 99% of the states have less than 1% users who have viewed advertisements published by the IRA. In terms

[17]The U.S Population data was gathered from United States Census Bureau. This data is publicly available. [23]

of clicks, all states have less than 1% users who interacted with these advertisement. In fact, more than 90% states have less than 0.05% users who interacted with these advertisements.

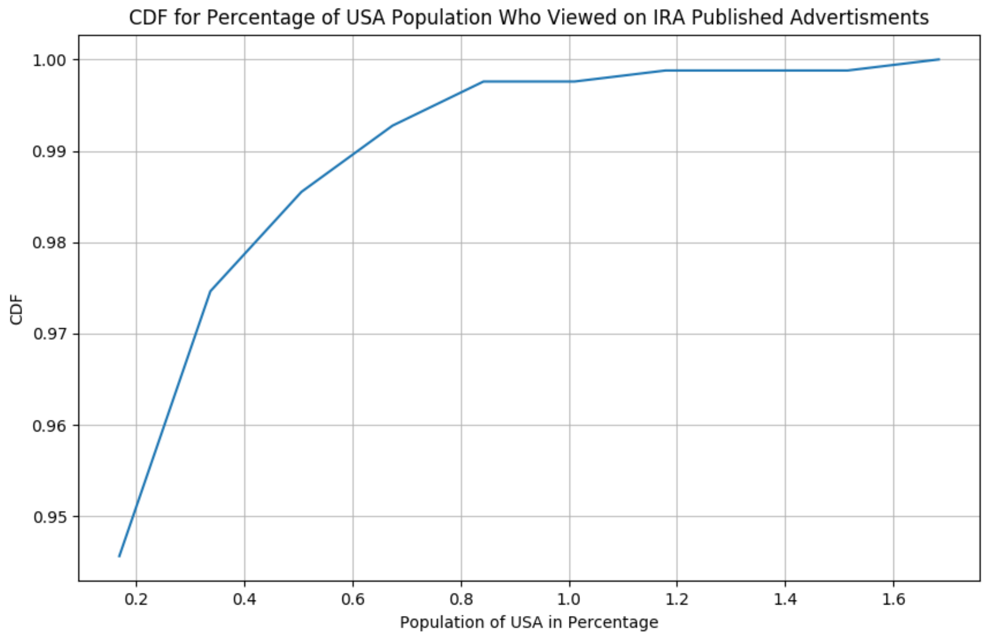


Figure 16: CDF of Audience Reached in Terms of Impressions

Since the US Elections are highly dependant on Swing States. Table 5 provides a description of Audiences Reached in the Swing States.

If the IRA campaign had been influential to any degree, it should have been able to : a) Influence undecided voters across USA, b) Have substantial influence on undecided voters in Swing States or c) Reduce the percentage of undecided voters across US or at a state level, particularly in the Swing States

First, according to a study by The Telegraph [6], the number of undecided voters in the 2012 election was approximately 3 Million (1% of the entire US Population). In 2016, this number doubled to 6 Million undecided voters (2% of the enitre US Population). As mentioned in Section 6.1, 0.2884% of the US population viewed the IRA published

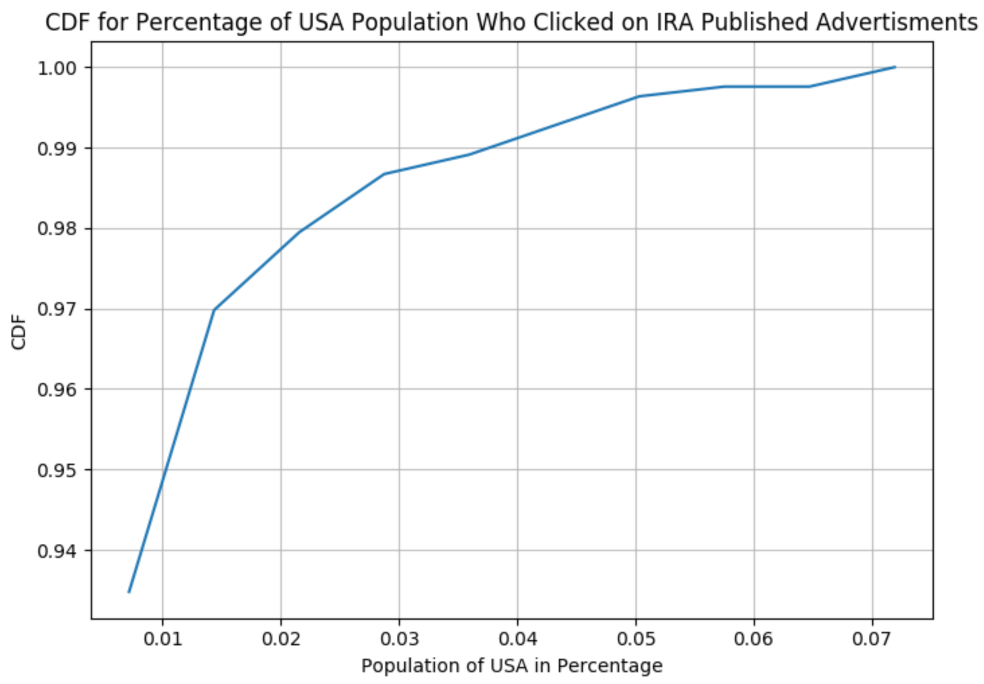


Figure 17: CDF of Audience Reached in Terms of Clicks

advertisements. 0.0159% of the US population interacted (clicked) with these advertisements. In addition, the number of undecided voters increase from 1% in 2012 to 2% in 2016 across USA.

Second, we look at the number of undecided voters in 2012 elections, 2016 elections and the Audience Reached for each Swing State in our dataset. This information is represented in Table 5. The Audience Reached (both in terms of impressions and clicks) is substantially lower than the number of undecided voters in each of the six swing states. In addition, the number of undecided voters increase from 2012 to 2016 in each of the swing states.

Finally, we cast light on our Twitter Mentions dataset. We observe only 1,90,295 users (excluding Twitter verified profiles and bots) interact with the IRA Trolls in form of mentions. This is a total of 0.05% of the USA population.

We can effectively conclude that a maximum audience reach of 1.6% in terms of

State	**No. Ads**	**2012 Undecided Voters**	**2016 Undecided Voters**	**Audience Reached (Impressions)**	**Audience Reached (Clicks)**
Ohio	168	1.6%	4.4%	0.0154339%	0.0011067%
Wisconsin	47	1.3%	5.2%	0.0041199%	0.0002797%
Florida	43	0.9%	3.1%	0.0060282%	0.0005916%
Michigan	36	1.1%	5.1%	0.0213606%	0.0009592%
Pennsylvania	20	1.4%	3.6%	0.0011784%	0.0000499%
Iowa	1	1.8%	6.0%	0.0002758%	0.0000002%

Table 5: Comparison of Audience Reached as per IRA Campaign and Undecided Voters in Swing States

impressions, 0.07% in terms of clicks across all US states and 0.05% of US population interacting with IRA Trolls on Twitter is not sufficient enough to have influenced the 2016 elections.

The IRA did try to influence the election by targeting individuals based on their location and personal interests. However, the campaign did not resolve to have an effective degree of influence on the elections.

6.4 Topical Relevance of Facebook Advertisements

In this Section, we first group the Facebook Interests in our dataset under topical umbrellas. Next, we measure the topical focus of these advertisements with respect to every state. Finally, we use Google Trends to measure the topical relevance of the advertisements in the geographically targeted locations.

6.4.1 Grouping of Facebook Interests

As mentioned in Section 5.4, we need to club the unique Facebook interests in our dataset according to their topical issues. In order to accomplish this task, we use **hierarchical clustering** to view the correlationship between each Facebook interest.

Each Facebook advertisement uses a group of Facebook Interests as a targeting parameter. For example, the interests used in one advertisement maybe "United States Senate, Election, Government, Politics". This means that these interests co-occur with each other at least once indicating that they are topically related to each other. To this extent, we generate a **co-occurrence** matrix. This is a square matrix of length 775 (total unique interests in our dataset). Each index [i][j] in the matrix represents the number of times Interest in row i has co-occurred with Interest in row j. The Interests are ordered alphabetically.

Next, we generate the **corelationship matrix** on the basis of the ranks in the co-occurrence matrix. We use this matrix to hierarchical cluster our Facebook Interests using scipy learning framework in Python. Our ultimate goal is to view the distance of separation between various interests and group the interests who are in close proximity to form topical groups. Scipy allows us to use various distance methods to view the correlationship between the interests [21]. We pick the **average distance method** as it provides us with the **highest cophenetic correlation coefficient** of 0.9665. The cophenetic correlation coefficient provides a measure of the quality of clustering. A value close to 1 represents a high quality clustering that preserves the pairwise distances in the correlationship matrix. A description of the various distance methods and their corresponding cophenetic correlation coefficients with respect to our dataset is provided in Table 6.

The **dendrogram** of the hierarchical clustering so obtained using the average distance method is presented in Figure 18. An *abbreviated version of the dendrogram* is presented in Figure 19. We use the lastp method provided by the scipy framework to abbreviate our dendrogram. This method only shows the last p merged clusters and abbreviates rest of the clusters as subsets of these p clusters. The value of p is optional. In a scenario where the value is not mentioned, the framework provides with a value of p to optimally visualise the abbreviated dendrogram [21].

Now, we have a dendrogram which represents the the correlationship distances between all Facebook interests in our dataset. We still need to group these interests into **topical clusters**. In order to do so, we need to pick a cut off distance in our dendrogram.

A standard method to pick a cut off distance is the **Elbow Method** [60]. This

Method	**Algorithm**	**Distance Formula** *(d(u,v)=)*	**Cophenetic Correlation Coefficient**
Single	Nearest Point Algorithm	$min(dist(u[i], v[j]))$	0.8476
Complete	Farthest Point Algorithm	$max(dist(u[i], v[j]))$	0.9141
Average	Unweighted Pair Group Method with Arithmetic Mean	$\sum_{ij} \frac{d([u], v[j])}{(\lvert u\rvert * \lvert v\rvert)}$	0.9665
Weighted	Weighted Pair Group Method with Arithmetic Mean	$\frac{d(s,v)+d(t,v)}{2}$	0.7401
Centroid	Unweighted Pair Group Method with Euclidean Distance	$\lvert\lvert c_s - c_t \rvert\rvert_2$	0.9612
Median	Weighted Pair Group Method with Euclidean Distance	$\sqrt{\lvert\lvert c_s - c_t \rvert\rvert_2}$	0.9060
Ward	Ward Variance Minimization Algorithm	$\sqrt{\frac{\lvert v\rvert+\lvert s\rvert}{T} d(v,s)^2 + \frac{\lvert v\rvert+\lvert t\rvert}{T} d(v,t)^2 - \frac{\lvert v\rvert}{T} d(s,t)^2}$	0.5864

Table 6: Description of Various Distance Methods and their Cophenetic Correlation Coefficients [21]

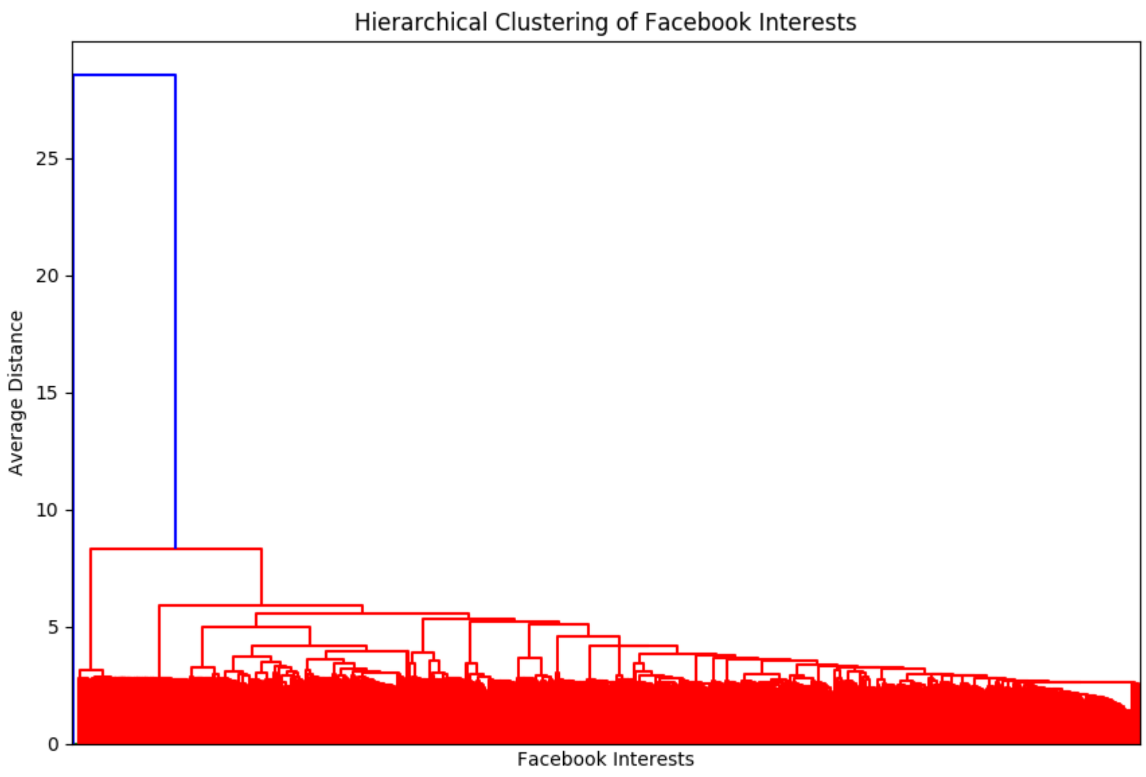


Figure 18: Dendrogram of Hierarchical Clustering using the Average Distance Method

method calculates the sum squared errors for every cut off distance on the dendrogram. On plotting these values, it is expected to obtain plot which has a sharp elbow. The X-Axis value corresponding to this elbow can be defined as the cut off for the dendrogram. The plot for Elbow Method is represented in Figure 20. The cut off distance so obtained is 2 units. However, one must note that this method is ambiguous and highly dependant on the dataset. Since our dataset is not very segregated, we cannot trust this value.

Another method to determine the dendrogram cut off is by using **Silhouette coefficients**. Silhouette coefficients ranges from -1 to +1. We obtain the silouhette coefficients for every possible dendrogram cut off. A high silhouette coefficient denotes optimal clustering. We plot the silhouette coefficients we obtain against every possible dendogram cut off in Figure 21. We note that if we cut the dendrogram at 5 units of average distance

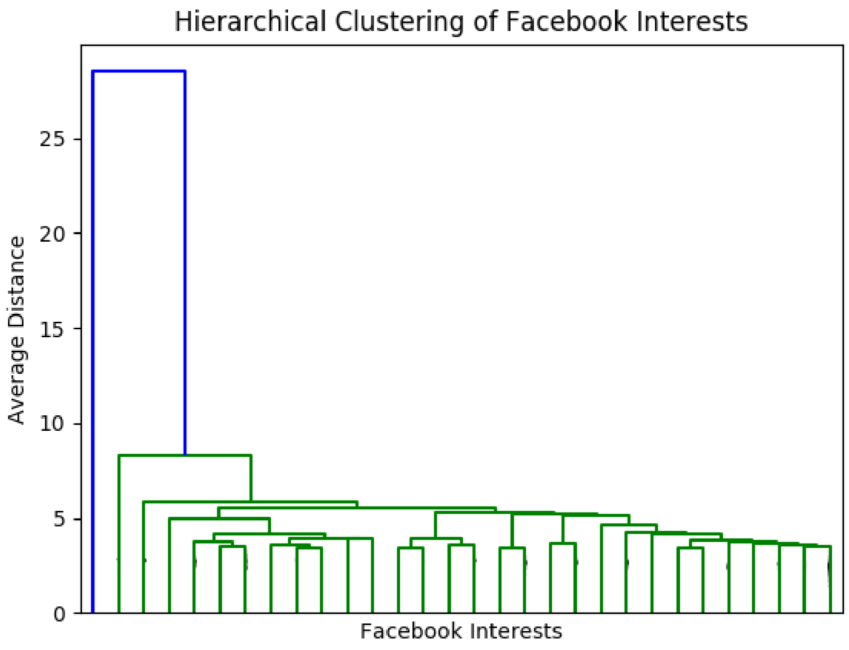


Figure 19: Compacted Dendrogram of Hierarchical Clustering using the Average Distance Method

(the max. silhouette coefficient), then we may obtain optimal clustering of Facebook interests into topical groups.

Selecting 5 units of average distance as the dendrogram cutoff, we generate a total of 8 topical groups (Group 0 - Group 7). These topical groups and their corresponding Facebook interests are listed in Section A.4. [18].

6.4.2 Topical Focus of Facebook Advertisements

We have already observed that advertisements drawn out by the IRA were targeted to users at a regional level using specific topical issues in each region. We have also noted that even though the efforts of micro-targeting were in the right direction, it did not amount to a substantial degree of influence in the USA 2016 elections. This makes

[18]We do note that Group 7 has a range of multiple topics. This is because these Interests are randomly used in multiple advertisements.

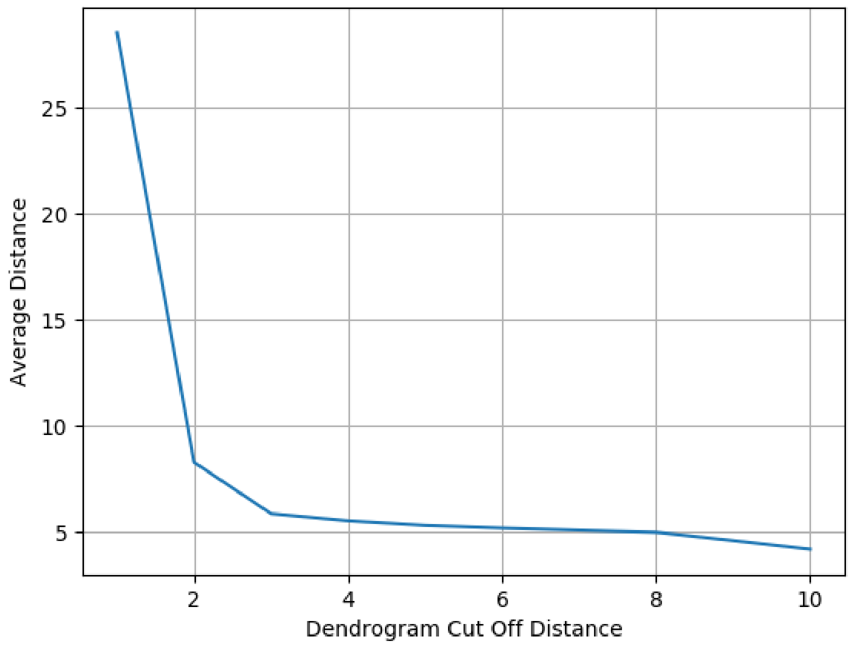


Figure 20: Elbow Method to obtain Dendrogram Cut Off Distance

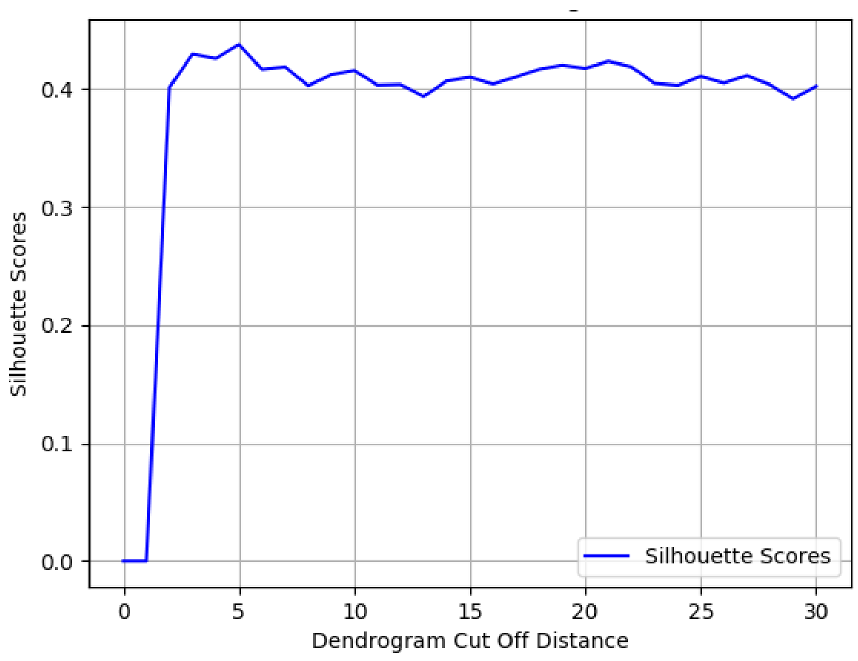


Figure 21: Silhouette Coefficients to obtain Dendrogram Cut Off Distance

us question where did the campaign fall short in order to gain a substantial degree of influence.

One of the main aspects of an advertisement campaign is that it should be topically concentrated to a handful topical issues. These issues should be relevant in the targeted geography. This assures that the campaign is topically focussed. In order to analyse the focus of a campaign in each state of USA, we use the metric Focus which was introduced in Section 5.5. We measure the focus of each state with respect to the clustered topical groups as well as individual interests in our dataset.

Clustered Topical Interests. In Section 6.4.1, we group the Facebook Interests into 8 topical groups. We want to see what was the topical focus of each state. We also keep a note of the focus values of each state.

We compute the focus values of each state as per Equation 5.4. We plot the values as a CDF presented in Figure 22. We note that while some states have a focus value of 0, some others have a focus value of 1. This is because these states either have no or very few advertisements (some states have only 1 advertisement). In our analysis we only look at states with at least 33 advertisements, which is the average number(33.86) of advertisements per state.

We note that the median of Focus from Figure 22 is 0.50. Only 7 out of 50 states have a focus value greater than 0.50 with a minimum of 33 advertisements. These states are Florida(0.76), Wisconsin(0.76), Illinois(0.73), New York(0.66), Texas(0.64), Washington(0.56) and Georgia(0.52). The Topical Interest Group of each state is listed in Table 7.

One may argue that the CDF in Figure 22 shows that over half of the states have a Focus value greater than 0.5. On the onset this may seem to be true. However, it seems so because a majority of the states have very few advertisements. Some states have as low as 1 advertisement. Naturally, these state may seem to have a substantial amount of focus. This is why we only look at states with at least 33 advertisements (average number of advertisements per state is 33.86) to make conclusions about the focus of the state.

State	**Topical Group of Focus**
Maryland	Group 3
Missouri	Group 3
Ohio	Group 3
New York	Group 7
Georgia	Group 3
California	Group 7
Texas	Group 7
Louisiana	Group 3
Illinois	Group 3
Virginia	Group 7
Wisconsin	Group 3
Florida	Group 3
Washington	Group 7
Michigan	Group 3
North Carolina	Group 7
Minnesota	Group 1
Pennsylvania	Group 7
Arizona	Group 7
Mississippi	Group 5
Alabama	Group 0
New Jersey	Group 7
New Mexico	Group 7
Kansas	Group 7
Oklahoma	Group 2
Tennessee	Group 7
Arkansas	Group 5
Massachusetts	Group 7
Connecticut	Group 7
Delaware	Group 7
Idaho	Group 4
Vermont	Group 7
Iowa	Group 5
Maine	Group 7
Nebraska	Group 5
Nevada	Group 7
New Hampshire	Group 7
Rhode Island	Group 7

Table 7: Topical Focus Group of USA States with Respect To Clustered Facebook Interests

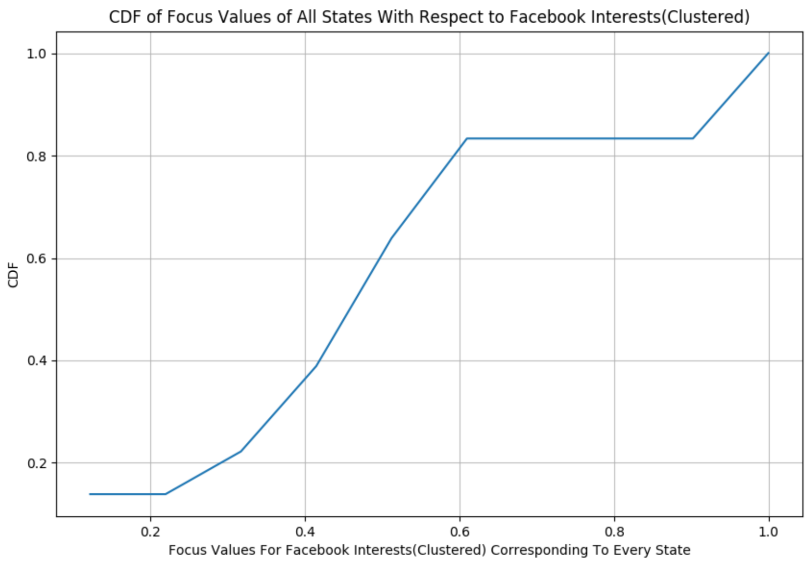


Figure 22: CDF of Focus Of Every State In USA With Respect to Clustered Topical Facebook Interests

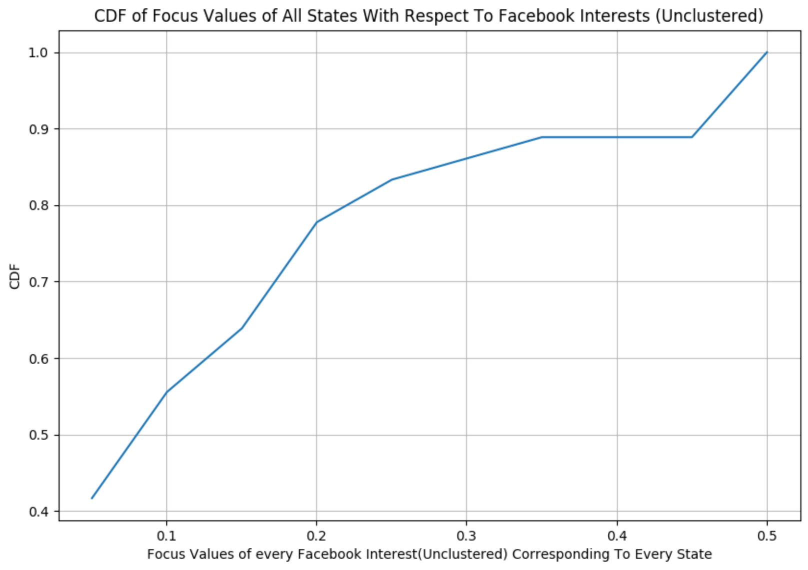


Figure 23: CDF of Focus Of Every State In USA With Respect to UnClustered Facebook Interests

It is further interesting to observe that 20 out of 38 states have their topical interest group as Group 7. Group 7 hosts a variety of topics which occur randomly with no topical pattern. This provides enough evidence to bolster our claim that there there was no topical focus at a state level to garner traction amongst audiences. This maybe one of the reasons why there was no substantial degree of influence in the IRA campaign.

UnClustered Topical Interests. We also plot the CDF of every state with respect to unclustered Facebook Interests. Figure 23 represents the plot. The median value of focus so obtained is 0.01. We note that only 10 out 50 states with a minimum of 33 advertisements have a focus value greater than 0.01. A listing of all states with their corresponding focus values and focus of interest can be found in Table 8.

Considering both clustered as well as unclustered interests, we note that only 7 states in case of clustered interests and 10 states in case of unclustered interests have a focus value greater than the median. In addition, we note that the topical focus in often a generic topic which may not be of supreme relevance in specific geographical regions. This leads us to conclude that the campaigns drawn out by the IRA did not have any topical objectivity. Even if they did, they were mostly focussed on generic topics which may not be of any relevance in the targeted region.

6.4.3 Topical Relevance

We observe that a range of interests were used to target various states in the IRA led campaign. We would want to estimate the relevance of these topics in the targeted geographical regions. An advertisement will only be able to influence audiences if the topic used to target the audience is of topical relevance locally.

In order to do, we use the notion of the digital truth serum introduce in Section 5.6. First, we rank the interests (clustered and unclustered) appearing in our Facebook Interests dataset with respect to their frequency. Second, we aggregate Google Trends score over the same period of time and rank these scores. Google Trends score provides us a measure of the actual interests amongst audiences. If there is a correlationship in the ranks of the interests in the IRA drawn campaign and Google Trends, it would mean

State	**Topical Group of Focus**	**Focus Value**
Maryland	Humanitarianism	0.2210
Missouri	Humanitarianism	0.2625
Ohio	Cop Block	0.2072
New York	Hispanic TV	0.0008
Georgia	Humanitarianism	0.0731
California	Antelope Valley College	0.0176
Texas	Independence	0.1724
Louisiana	Humanitarianism	0.1578
Illinois	Cop Block	0.0033
Virginia	Southern United States	0.0188
Wisconsin	Humanitarianism	0.1204
Florida	Martin Luther King Jr.	0.0235
Washington	Independence	0.0076
Michigan	Police brutality in the United States	0.0051
North Carolina	Confederate Flag	0.0072
Minnesota	Islam	0.0160
Pennsylvania	HispanicTV	0.0666
Arizona	Bisexuality	0.0447
Mississippi	U.S. Patriot Tactical	0.0108
Alabama	U.S. Patriot Tactical	0.0098
New Jersey	Hispanic TV	0.0116
New Mexico	Veterans	0.0200
Kansas	LGBT rights by country	0.1166
Oklahoma	Texas Secession	0.0465
Tennessee	Confederate Flag	0.1000
Arkansas	U.S. Patriot Tactical	0.0454
Massachusetts	Liberalism	0.1428
Connecticut	Hispanic TV	0.1666
Delaware	Hispanic TV	0.3333
Idaho	Conservatism	0.1666
Vermont	Hispanic TV	0.1666
Iowa	U.S. Patriot Tactical	0.0833
Maine	Hispanic TV	0.5000
Nebraska	U.S. Patriot Tactical	0.0833
Nevada	Hispanic TV	0.5000
New Hampshire	Hispanic TV	0.5000
Rhode Island	Hispanic TV	0.5000

Table 8: Topical Focus Group of USA States with Respect To UnClustered Facebook Interests

that the IRA had actually used relevant topics to target the states.

Correlation across USA. We rank the interests (unclustered) in our dataset as well as Google Trends (across USA). We use Spearman's rank correlation to draw the correlation between the actual interests of audiences across USA (Google Trends) and that in our dataset. *We report a correlation coefficient of 0.1292 with a very low P value of 0.0003.*

A correlation value of 0.1292 means that there was little or no correlationship between the actual local interests and the interests used in the campaign drawn out by the IRA. In addition, a low P Value (usually $p<0.05$) indicates high reconstruction quality of results. We report a very low p value - 0.0003.

We can conclude that the interests used to target audiences did not truly represent the actual geographical interests. This can be a strong reason why the advertisements laid out by the IRA did not influence sufficient audiences.

We further look at the topical relevance at a state level. We first analyse the relevance using unclustered interests. Finally, we analyse the relevance for focussed topical groups.

Topical Relevance at State Level for Unclustered Interests. We rank the Facebook interests used as a targeting parameter for every state. Correspondingly, we rank the Google Trends of these interests in every state. We represent the correlation coefficients of each state as a CDF in Figure 25. We observe that the maximum probability of the correlation coefficient is 0.55 across all states. This means that there is negligible (highly negative correlation in over 40% states) correlation between the interests used to target the states and the actual interests amongst the audiences.

Topical Relevance at State Level for Focused Topical Groups. We repeat the same process of correlating Google Trends in each state with interests used to target the states. But, in this case, we use the focussed topical groups of each state (we obtained the focussed topical groups in Section 6.4.2). The correlation coefficients of each state is represented as a CDF in Figure 24. We note that there is a negative correlationship between the focused topic and the actual geographical interests.

We report a negative correlation at a state level as well as across the entire country.

This provides us enough evidence to conclude that the IRA did not use relevant interests to target users. The interests used by the IRA did not truly reflect the concerns of the society at large. This can be major reason why the campaign drawn out by the IRA did not have a substantial degree of influence.

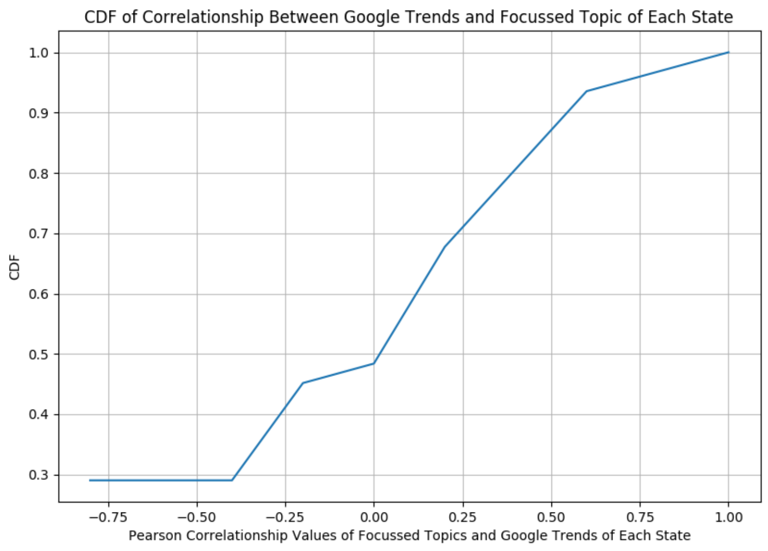


Figure 24: CDF of Correlation of Focussed Topics and Google Trends in Each State

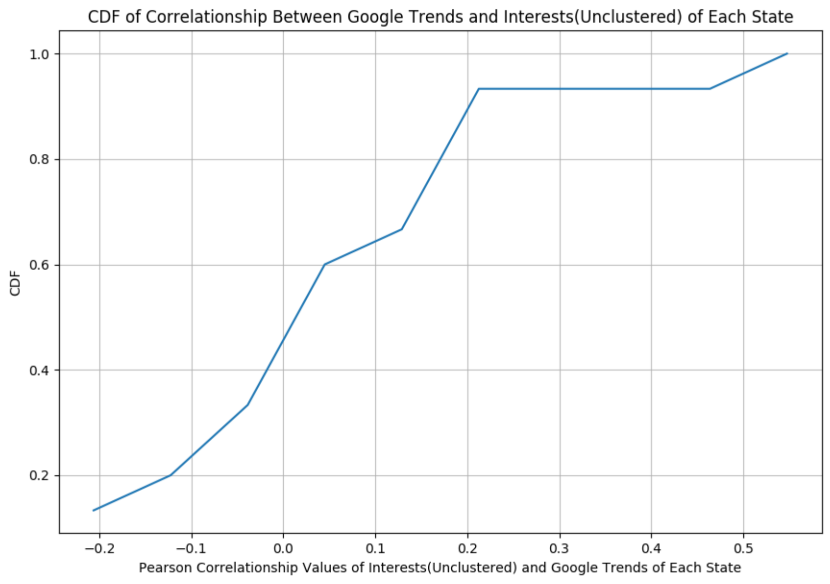


Figure 25: CDF of Correlation of Facebook Interest(Unclustered) and Google Trends in Each State

7 Conclusion

In this Section, we first present an overview of the outcomes of this project. We conclude this project by throwing light in the future direction of research in this domain.

7.1 Key Findings

This project casts light on the efforts of state sponsored advertisement campaigns to influence the presidential elections in USA held in 2016. We show that the the IRA micro-targeted users with respect to their geographic locations. Further, they also use personal interests of users as a targeting parameter. However, the outcome of the campaign was not substantial. We investigate why the campaign failed to garner sufficient influence amongst audiences. We arrive to a conclusion that the advertisements did not have any topical objectivity. In the states where the campaign was focussed, the campaign did not address relevant topical issues.

- **Users Were Targeted Using Topical Interests According To Their Geographic Locations.** We discover that users were targeted geographically. Personal interests of users were exploited to act as a targeting parameter. Different states were targeted with varying topical interests.

- **The Resulting Degree of Influence Was Not Substantial To Influence The Elections.** We discover that the resulting degree of influence was substantially less than the percentage of undecided voters in every swing state. We also note that the number of undecided voters actually increase from 2012 to 2016 across USA as well as across every swing state. This indicates that the IRA campaign did not garner enough momentum to have influenced the elections.

- **Campaigns Did Not Have Any Topical Objectivity.** The advertisements were not topically focussed in a majority of the targeted states. The advertisements ranged over a number of topics with no clear topical pattern. The randomness of these advertisements can be one of the reasons why these campaigns did not garner a substantial following.

- **Topical Issues Used To Target Audiences Did Not Echo The Actual Interests Of The Society.** The topical interests used to target users did not resemble the actual topical interests of the users in these geographical regions. This is one of the main reasons why users may not have been interested in these advertisements in the first place.

7.2 Future Work

The findings of this project can lay the foundation for future research in this domain.

Political Communities In The Twitter Ecosystem. In this project, we majorly focus on Facebook advertisements. In a preliminary analysis, the Twitter Trolls dataset shows that multiple communities interact with the IRA trolls. These communities are displayed in Figure 26. It would be very interesting to analyse the alignment of political conversations in these communities. If there are definite structures in which state sponsored trolls spawn, social network companies such as Facebook and Twitter can benefit by being aware of these structures and monitoring them.

Micro-Targeted Marketing. Digital advertisements allow marketeers to publish micro-targeted advertisements. It is claimed that these advertisement are far superior to physical advertisements because of the extreme personal reach they have. However, we observe that that such advertisements do not amount to have a substantial degree of influence in our project. In addition, the benchmark data analysing such advertisements across USA reveals that users hardly interact with such advertisements. Future research can question the suitability of such advertisements taking into account that users have digitally educated themselves over time and may view such advertisements in the ranks of spam E-Mail.

Facebook Interests and User Privacy. Facebook Interests are assigned to users according to their communication patterns on the social media platform. Facebook does not provide transparency on how these interests are allocated. Future research can question the validity of such interests and survery the precautions taken by Facebook to safeguard users in case of breaches.

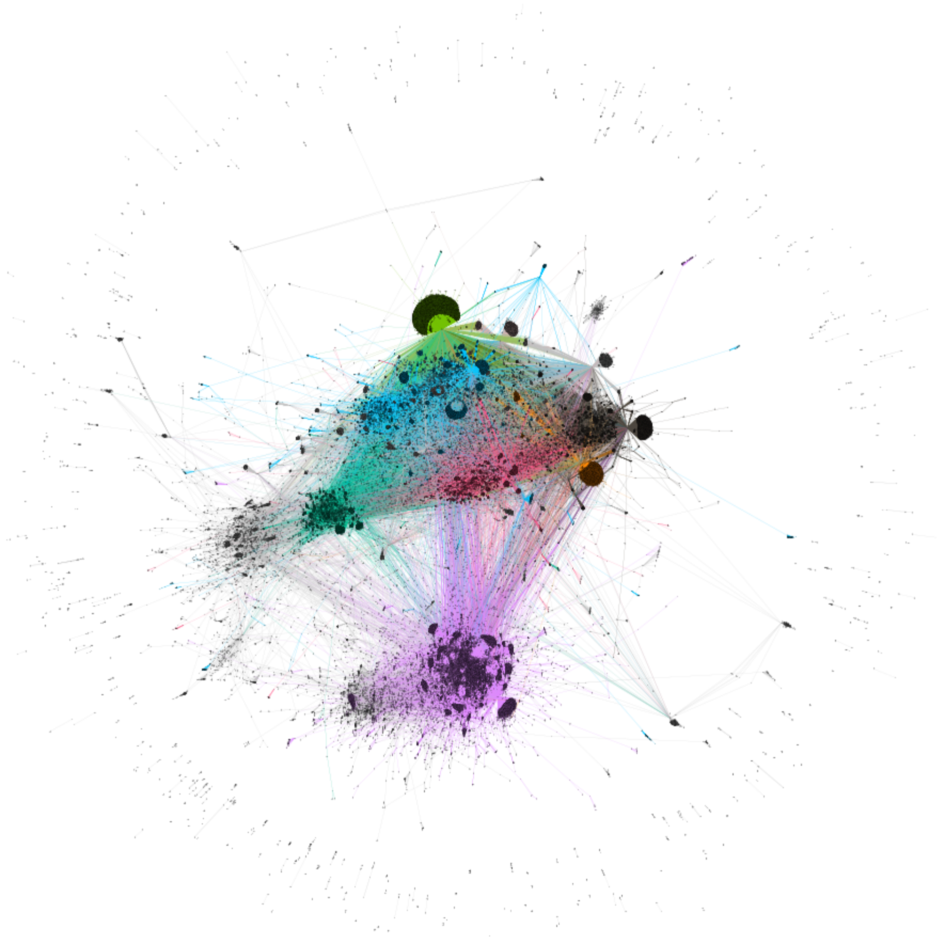


Figure 26: Prelimnary Results Showing Distribution of Communities On Twitter

References

[1] About boosting posts. https://www.facebook.com/business/help/240208966080581. Accessed : 08/08/2018.

[2] About detailed targeting. https://www.facebook.com/business/help/182371508761821. Accessed : 07/08/2018.

[3] About facebook check-ins. https://www.facebook.com/business/help/1490758697831305. Accessed : 07/08/2018.

[4] About location services. https://www.facebook.com/location07/08/2018.

[5] Advertising objectives. https://www.facebook.com/business/help/197976123664242. Accessed : 08/08/2018.

[6] America's undecided voters: could they swing the whole 2016 election? https://www.theguardian.com/us-news/datablog/2016/nov/03/undecided-voters-election-2016-hillary-clinton-donald-trump. Accessed : 18/8/18.

[7] Cia salaries in washington, dc , us area. https://www.glassdoor.co.uk/Salary/CIA-Washington-DC-Salaries-EI_IE41381.0,3_IL.4,17_IM911.htm. Accessed : 23/8/18.

[8] The complete resource to understanding facebook ads cost – 2017 benchmarks! https://adespresso.com/blog/facebook-ads-cost/. Accessed : 17/08/2018.

[9] Death penalty database - afghanistan. https://www.deathpenaltyworldwide.org/country-search-post.cfm?country=Afghanistan. Accessed : 08/08/2018.

[10] Encourage customers to buy with the aida model. https://www.winstmagneet.nl/klanten-verleiden-het-aida-model/. Accessed : 17/8/18.

[11] Facebook ad benchmarks for your industry [new data]. https://www.wordstream.com/blog/ws/2017/02/28/facebook-advertising-benchmarks. Accessed : 17/08/2018.

[12] Facebook advertising costs by industry 2018. https://fitsmallbusiness.com/how-much-does-facebook-advertising-cost/. Accessed : 17/08/2018.

[13] Facebook advertisment dataset. https://democrats-intelligence.house.gov/social-media-content/social-media-advertisements.htm. Accessed : 14/08/2018.

[14] Facebook is banning hundreds more accounts run by russian trolls. https://www.recode.net/2018/4/3/17194430/facebook-bans-ira-russian-troll-accounts. Accessed : 12/08/2018.

[15] Facebook is banning hundreds more accounts run by russian trolls. https://newsroom.fb.com/news/2017/09/information-operations-update/. Accessed : 12/08/2018.

[16] Google says russia tried to influence us election using adverts on youtube and gmail. https://www.independent.co.uk/news/world/americas/google-russia-us-election-adverts-found-youtube-gmail-donald-trump-president-investigation-latest-a7990546.html. Accessed : 12/08/2018.

[17] Information and operations facebook. https://fbnewsroomus.files.wordpress.com/2017/04/facebook-and-information-operations-v1.pdf. Accessed : 12/08/2018.

[18] Russian influence campaign sought to exploit americans' trust in local news. https://www.npr.org/2018/07/12/628085238/russian-influence-campaign-sought-to-exploit-americans-trust-in-local-news?t=1534550843985. Accessed : 17/8/18.

[19] Russian spies adopt new tactics to battle old enemy. https://www.ft.com/content/ae1cd0c2-c170-11e6-9bca-2b93a6856354. Accessed : 23/8/18.

[20] Russia's influence campaign targeting the 2016 us presidential election. https://www.dni.gov/files/documents/ICA_2017_01.pdf. Accessed : 21/8/18.

[21] scipy linkage matrix distances. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hiera
Accessed : 20/8/18.

[22] St petersburg 'troll farm' had 90 dedicated staff working to influence us election campaign. https://www.independent.co.uk/news/world/europe/russia-us-election-donald-trump-st-petersburg-troll-farm-hillary-clinton-a8005276.html. Accessed : 11/08/2018.

[23] State population totals and components of change: 2010-2017. https://www.census.gov/data/datasets/2017/demo/popest/state-total.html. Accessed : 18/8/18.

[24] Twitter is testing a big change: Doubling the length of tweets from 140 to 280 characters. https://www.recode.net/2017/9/26/16364002/twitter-longer-tweets-character-limit-140-280. Accessed : 09/08/2018.

[25] Twitter's list of 2,752 russian trolls. https://www.recode.net/2017/11/2/16598312/russia-twitter-trump-twitter-deactivated-handle-list. Accessed : 11/08/2018.

[26] Update on twitter's review of the 2016 u.s. election. https://blog.twitter.com/official/en_us/topics/company/2018/2016-election-update.html. Accessed : 12/08/2018.

[27] Usa census data. https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?src=b
Accessed : 17/08/2018.

[28] Where is the centre of london? http://www.bbc.co.uk/london/content/articles/2005/08/15/charingcros
feature.shtml. Accessed : 08/08/2018.

[29] Exposing russia's effort to sow discord online: The internet research agency and advertisements, February 2018. https://democrats-intelligence.house.gov/social-media-content. Accessed : 07/08/2018.

[30] Pablo Barberá, John T Jost, Jonathan Nagler, Joshua A Tucker, and Richard Bonneau. Tweeting from left to right: Is online political communication more than an echo chamber? *Psychological science*, 26(10):1531–1542, 2015.

[31] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. 2016.

[32] Shweta Bhatt, Sagar Joglekar, Shehar Bano, and Nishanth Sastry. Illuminating an ecosystem of partisan websites. *arXiv preprint arXiv:1803.03576*, 2018.

[33] Anders Brodersen, Salvatore Scellato, and Mirjam Wattenhofer. Youtube around the world: geographic popularity of videos. In *Proceedings of the 21st international conference on World Wide Web*, pages 241–250. ACM, 2012.

[34] José González Cabañas, Ángel Cuevas, and Rubén Cuevas. Facebook use of sensitive data for advertising in europe. *arXiv preprint arXiv:1802.05030*, 2018.

[35] Michael D Conover, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Partisan asymmetries in online political activity. *EPJ Data Science*, 1(1):6, 2012.

[36] Stefan Des. The history of facebook ads – how facebook advertising evolved in the last 13 years. https://leadsbridge.com/infographic-history-facebook-ads/. Accessed : 07/08/2018.

[37] Emilio Ferrara. Disinformation and social bot operations in the run up to the 2017 french presidential election. 2017.

[38] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Reducing controversy by connecting opposing views. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 81–90. ACM, 2017.

[39] Yanbo Ge, Christopher R Knittel, Don MacKenzie, and Stephen Zoepf. Racial and gender discrimination in transportation network companies. Technical report, National Bureau of Economic Research, 2016.

[40] Aniko Hannak, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. Measuring price discrimination and steering on e-commerce web sites. In *Proceedings*

of the 2014 conference on internet measurement conference, pages 305–318. ACM, 2014.

[41] Anikó Hannák, Claudia Wagner, David Garcia, Alan Mislove, Markus Strohmaier, and Christo Wilson. Bias in online freelance marketplaces: Evidence from taskrabbit and fiverr. In *CSCW*, pages 1914–1933, 2017.

[42] Simon Hegelich and Dietmar Janetzko. Are social bots on twitter political actors? empirical evidence from a ukrainian social botnet. In *ICWSM*, pages 579–582, 2016.

[43] Sarah Kessler. The history of advertising on facebook. https://mashable.com/2011/06/28/facebook-advertising-infographic/. Accessed : 07/08/2018.

[44] Young Mie Kim, Jordan Hsu, David Neiman, Colin Kou, Levi Bankston, Soo Yun Kim, Richard Heinrich, Robyn Baragwanath, and Garvesh Raskutti. The stealth media? groups and targets behind divisive issue campaigns on facebook. *Political Communication*, pages 1–29, 2018.

[45] Aleksandra Korolova. Privacy violations using microtargeted ads: A case study. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 474–482. IEEE, 2010.

[46] Srijan Kumar, Justin Cheng, Jure Leskovec, and VS Subrahmanian. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 857–866. International World Wide Web Conferences Steering Committee, 2017.

[47] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 79–84. acm, 2012.

[48] Fred Morstatter, Yunqiu Shao, Aram Galstyan, and Shanika Karunasekera. From alt-right to alt-rechts: Twitter analysis of the 2017 german federal election. In

Companion of the The Web Conference 2018 on The Web Conference 2018, pages 621–628. International World Wide Web Conferences Steering Committee, 2018.

[49] Casey Newton. Why we can't trust facebook's story about russian ads. https://www.theverge.com/2017/9/8/16277144/facebook-russian-ads-political-explainer-credibility. Accessed : 12/08/2018.

[50] Jessica Guynn Nick Penzenstadler, Brad Heath. We read every one of the 3,517 facebook ads bought by russians. here's what we found, May 2018. https://eu.usatoday.com/story/news/2018/05/11/what-we-found-facebook-ads-russians-accused-election-meddling/602319002/. Accessed : 10/08/2018.

[51] Arthur F Peterson. *Pharmaceutical selling," detailing," and sales training.* Heathcote-Woodbridge, 1959.

[52] Marian-Andrei Rizoiu, Timothy Graham, Rui Zhang, Yifei Zhang, Robert Ackland, and Lexing Xie. # debatenight: The role and influence of socialbots on twitter during the 1st us presidential debate. *arXiv preprint arXiv:1802.09808*, 2018.

[53] Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Alessandro Flammini, and Filippo Menczer. The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*, 2017.

[54] Experian Simmons. Simmons oneview faq, 2015.

[55] Kate Starbird. Examining the alternative media ecosystem through the production of alternative narratives of mass shooting events on twitter. In *ICWSM*, pages 230–239, 2017.

[56] Seth Stephens-Davidowitz and Steven Pinker. *Everybody lies: big data, new data, and what the internet can tell Us about who we really are.* HarperCollins New York, 2017.

[57] Edward K Strong. The psychology of selling and advertising. new york [etc.]. *McGraw-Hill book company, inc.[online]*, 462:9, 1925.

[58] Onur Varol, Emilio Ferrara, Clayton A Davis, Filippo Menczer, and Alessandro Flammini. Online human-bot interactions: Detection, estimation, and characterization. *arXiv preprint arXiv:1703.03107*, 2017.

[59] Samuel C Woolley and Douglas R Guilbeault. Computational propaganda in the united states of america: Manufacturing consensus online. *Computational Propgaganda Research Project*, page 22, 2017.

[60] Antoine E Zambelli. A data-driven approach to estimating the number of clusters in hierarchical clustering. *F1000Research*, 5, 2016.

[61] Savvas Zannettou, Tristan Caulfield, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini, and Jeremy Blackburn. Disinformation warfare: Understanding state-sponsored trolls on twitter and their influence on the web. *arXiv preprint arXiv:1801.09288*, 2018.

A Appendix

A.1 Facebook Interests and User Locations

User Interests. The most important parameter used by Facebook to serve advertisements is known as **Interests**. Interests are assigned by Facebook to users based upon their navigation patterns on Facebook. Facebook assigns interests to users based upon the following :

- Applications they interact with
- Advertisements they click on
- Pages they interact with
- Engagement Patterns "in and off Facebook" with respect to online shopping behaviour, travel interests or intent
- Basic demographic data such as gender, age and location
- Device and device network that is being used to navigate Facebook

Facebook assigns interests to users based on these criteria. Further, marketers can use these interests to target advertisements to groups of people interested in these topic. Marketeers can also exclude audiences interests in certain topics [2].

Users can view the Interests they have been assigned on Facebook and the reason they have been assigned that interest. A screenshot can be seen in Figure 27 where a users interests and the reason of assignment of such interest is shown on Facebook. Users have the privilege to remove interests that they may feel aptly does not describe them. However, this is not an advertised feature and many users are unaware of this option.

User Location. Facebook keeps a track of location of users. According to Facebook, it builds a log of all locations that the user has travelled to. Facebook uses location services which is a very common feature readily available in smartphones. In addition, it also uses location services via browsers. Facebook also uses other features on their

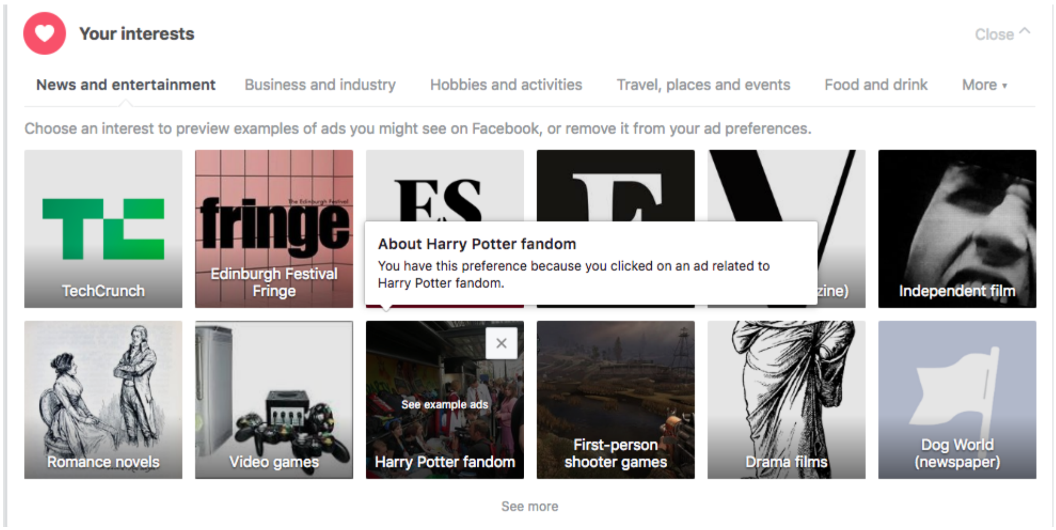

Your interests
Close
News and entertainment
Business and industry
Hobbies and activities
Travel, places and events
Food and drink
More
Choose an interest to preview examples of ads you might see on Facebook, or remove it from your ad preferences.
TechCrunch
Edinburgh Festival Fringe
About Harry Potter fandom
You have this preference because you clicked on an ad related to Harry Potter fandom.
Independent film
Romance novels
Video games
See example ads
Harry Potter fandom
First-person shooter games
Drama films
Dog World (newspaper)
See more

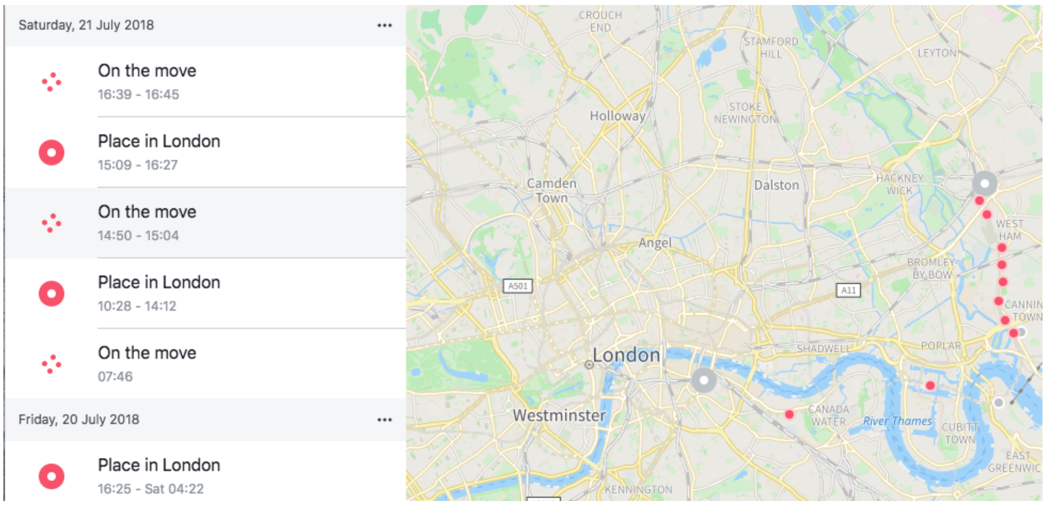
Figure 27: User Interests on Facebook

platform to track location of users. Check-Ins and geolocation of photos are two such features [4][3].

The location history log is a comprehensive description of all movements by the user at a particular date and time when location services could be accessed. Facebook classifies the location of a user into two categories : a) Place Visited and b) "On The move" - This describes the route taken by the user while travelling. User can view their location history under Privacy in Settings. A snapshot of a location history is provided in Figure 28. The red path indicated on the map highlights the path taken by the user on July 21, 2018 in between 14:50 and 15:04 hours. Facebook provides users the autonomy to delete all location history or even delete the history for particular days. However, this feature is not marketed to users and many users may not even be aware of the same.

Facebook uses the location history of users to target users advertisments. Facebook provides marketeers the option to target users by location. This is a dynamic feature and marketeers can target users by location using the following combinations :

- Everyone in the specified locations - This includes users who live, travel or even passing through the mentioned locations
- Users who reside in the specifed locations- This includes users whose residences are


Saturday, 21 July 2018

On the move
16:39 - 16:45

Place in London
15:09 - 16:27

On the move
14:50 - 15:04

Place in London
10:28 - 14:12

On the move
07:46

Friday, 20 July 2018

Place in London
16:25 - Sat 04:22

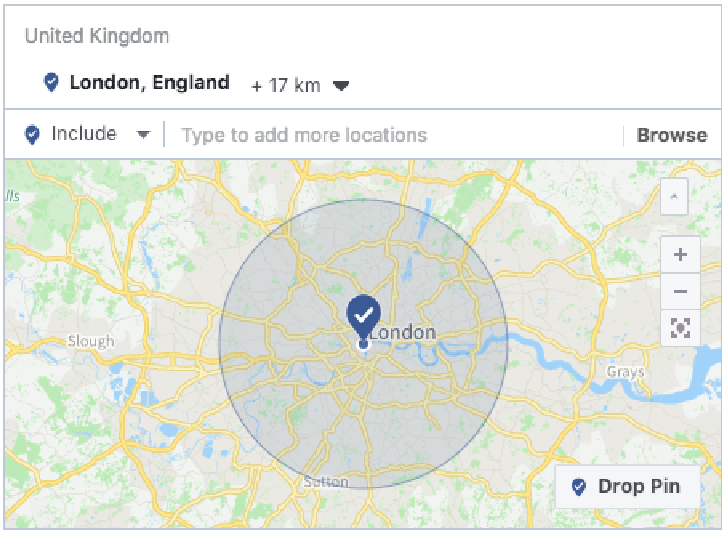
Figure 28: Location History of a user on Facebook

in the range of the specified locations

- Users who were recently in the specifed locations - This includes users whose most recent location in their location history is the specified locations
- Users who are travelling in the specified locations - This includes users whose most recent location in their location history is the specified locations. In addtion, their location of residence needs to be 200 Km away from the specified locations

Marketeers can add multiple locations. Facebook provides the option to target particular states, cities in countries. In addition, marketeers may also mention the radius of location in which they are interested in targeting the advertisments. For example, one can select that the advertisment must be displayed to individuals who in a range of 17 Km in London. This range is usually measured from the centre of the city. Hence, the advertisments ranging 17 Km in London would mean that the advertiments will be displayed to users who are within a radius of 17 Km from Charring Cross [28]. A representation of the same is present in Figure 29.

While Facebook's record of history of user location is essential for Facebook to serve advertisments to users, on the flipside it can be highly vulnerable. Security breaches may lead to hackers hackers obtaining potenially sensitive data. There have been known cases


United Kingdom

London, England + 17 km

Include | Type to add more locations | Browse

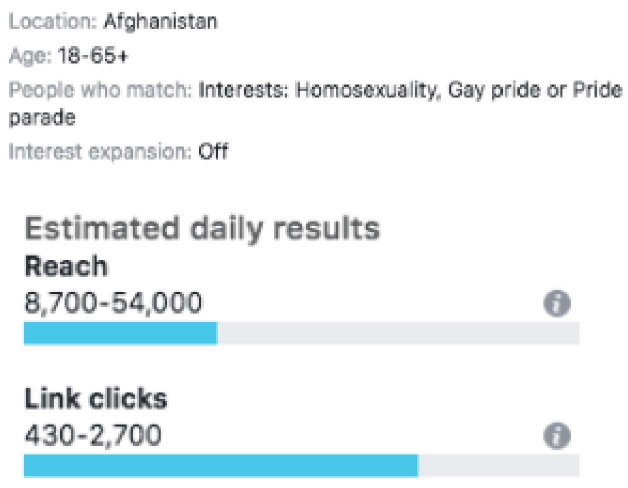
Figure 29: Facebook Advertisment targetting a radius of 17Km from the centre of London

of potential attacks where the location of history of users could have been compromised [45]. Discriminative advertisments can be published based on location of the user. Data gathered from response to advertisments can be used in a potentially harmful way. For example, it is illegal to be homosexual in many countries. In fact, the punishment for being homosexual is the capital punishment [9]. Malicious actors could phish for users who are present in Afghanistan and interested in homosexuality and blackmail them.

To enumerate on our example of phising, we set up an advertisment on Facebook which targets people who are living in Afghanistan and interested in Homosexuality, Gay Pride or the Pride Parade. Facebook allows us to move forward with this combination on August 8, 2018. Facebook estimates that around 430 - 2,700 individuals will click on our advertisment for just GBP 8 per day. We do not run the advertisments. The screenshots are presented in Figure 30.

A.2 Description Of Facebook Advertisment Campaign Objectives

The Objectives of various types of Facebook Advertisment Campaigns are listed in Table 9.



Location: Afghanistan
Age: 18-65+
People who match: Interests: Homosexuality, Gay pride or Pride parade
Interest expansion: Off

Estimated daily results
Reach
8,700-54,000

Link clicks
430-2,700

Figure 30: Facebook Advertisment depicting reach and clicks received on advertisment targetted to homosexuals in Afghanistan

A.3 High Level View of The Database Schema

Facebook Advertisments Database Schema. The Facebook Advertisments Database Schema (High Level) is presented in Figure 31.

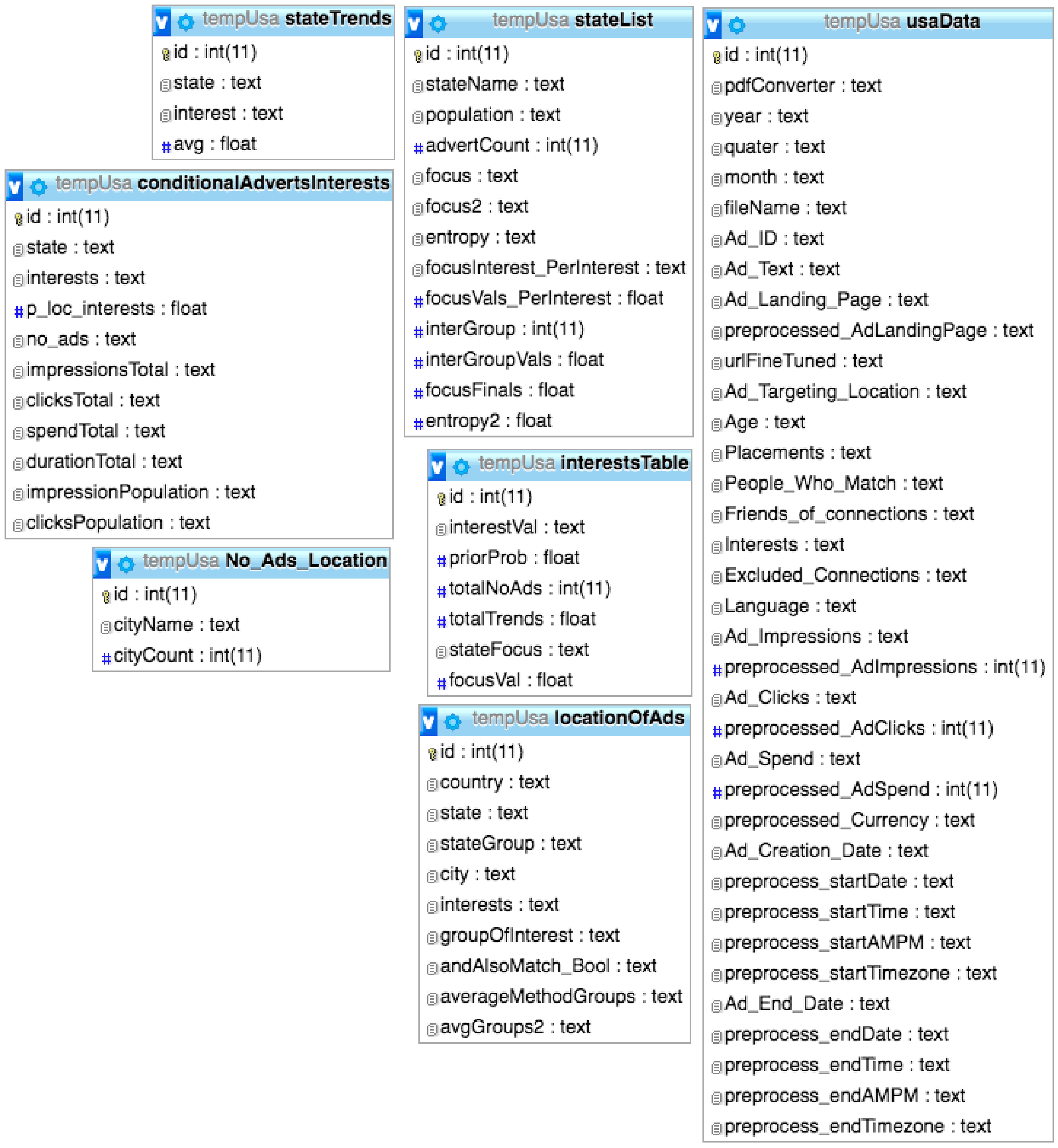
Twitter Trolls Database Schema. The Twitter Trolls Database Schema (High Level) is presented in Figure 32.

A.4 Facebook Topical Groups

This Section presents a listing of the topical groups that have been obtained by Hierachially clustering Facebook Interests.

Group 0. Humans of New York,Martin Luther King Jr.,New York City,Reading

Group 1. Abbas ibn Ali,Abraham in Islam,Abu Bakr,Abu Talib ibn AbduI-Muttalib,Ahl aI-Bayt,Ali,Ali aI-Asghar ibn Husayn,Ali ibn Husayn Zayn al-Abidin, AliaI-Ridha,As-salamu alaykum,Basmala,Caliphate,Eid al-Adha,Eid al-Fitr,Employers Allah,Fard,Fatimah,Fiqh,Glossary of Islam,Hadith,Hajj,Hasan aI-Askari,Hasan ibn Ali, Hijra (Islam),Husayn ibn Ali,Imam,Imam Ali Mosque,Ja'far aI-Sadiq,Jesus in Islam,Kaaba,KARBALA IMAM HUSSEIN (A.S)



tempUsa stateTrends
id : int(11)
state : text
interest : text
avg : float

tempUsa conditionalAdvertsInterests
id : int(11)
state : text
interests : text
p_loc_interests : float
no_ads : text
impressionsTotal : text
clicksTotal : text
spendTotal : text
durationTotal : text
impressionPopulation : text
clicksPopulation : text

tempUsa No_Ads_Location
id : int(11)
cityName : text
cityCount : int(11)

tempUsa stateList
id : int(11)
stateName : text
population : text
advertCount : int(11)
focus : text
focus2 : text
entropy : text
focusInterest_PerInterest : text
focusVals_PerInterest : float
interGroup : int(11)
interGroupVals : float
focusFinals : float
entropy2 : float

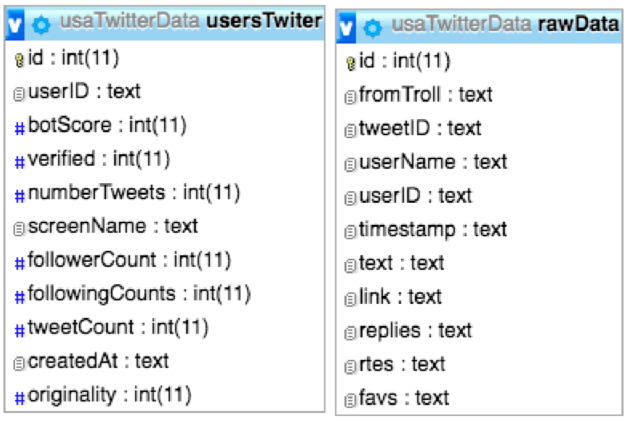
tempUsa interestsTable
id : int(11)
interestVal : text
priorProb : float
totalNoAds : int(11)
totalTrends : float
stateFocus : text
focusVal : float

tempUsa locationOfAds
id : int(11)
country : text
state : text
stateGroup : text
city : text
interests : text
groupOfInterest : text
andAlsoMatch_Bool : text
averageMethodGroups : text
avgGroups2 : text

tempUsa usaData
id : int(11)
pdfConverter : text
year : text
quater : text
month : text
fileName : text
Ad_ID : text
Ad_Text : text
Ad_Landing_Page : text
preprocessed_AdLandingPage : text
urlFineTuned : text
Ad_Targeting_Location : text
Age : text
Placements : text
People_Who_Match : text
Friends_of_connections : text
Interests : text
Excluded_Connections : text
Language : text
Ad_Impressions : text
preprocessed_AdImpressions : int(11)
Ad_Clicks : text
preprocessed_AdClicks : int(11)
Ad_Spend : text
preprocessed_AdSpend : int(11)
preprocessed_Currency : text
Ad_Creation_Date : text
preprocess_startDate : text
preprocess_startTime : text
preprocess_startAMPM : text
preprocess_startTimezone : text
Ad_End_Date : text
preprocess_endDate : text
preprocess_endTime : text
preprocess_endAMPM : text
preprocess_endTimezone : text

Figure 31: Facebook Advertisments Database Schema

SHRINE,Mary in Islam, Mecca,Medina,Mufti,Muhammad aI-Baqir,Muhammad aI-Mahdi,Musa al-Kadhim,Names of God in Islam,Prophet,Prophets and messengers in Islam,Ramadan,Sakinah (Fatima aI-Kubra) bint Husayn, Shahada,Sharia,Sheikh,Shia Islam,Sukayna bint Husayn,Sunnah,Takbir,Twe



usaTwitterData usersTwiter
id : int(11)
userID : text
botScore : int(11)
verified : int(11)
numberTweets : int(11)
screenName : text
followerCount : int(11)
followingCounts : int(11)
tweetCount : int(11)
createdAt : text
originality : int(11)

usaTwitterData rawData
id : int(11)
fromTroll : text
tweetID : text
userName : text
userID : text
timestamp : text
text : text
link : text
replies : text
rtes : text
favs : text

Figure 32: Twitter Trolls Database Schema

Hussain,Zakat,Zaynab bint AIi

Group 2. Daughters of the Republic of Texas,Don't Mess with Texas,Don't Mess with Texas Program,Hog Hunting Texas Style,Houston,Keep Texas Working, made in texas,Republic of Texas,Texas,Texas Born,Texas Deer Association,Texas Deer Hunting,Texas Got It Right,Texas Gun Owner,Texas Gun Talk,Texas Hog Hunting, Texas Nationalist Movement,Texas Parks and Wildlife - Hunt,Texas Proud,Texas Secession,Texas secession movements,Texas Values,The Texas Huntress

Group 3. 2016,Active Self Protection,Administrative law,African American,African American Museum in Philadelphia,African culture,African Methodist Episcopal Church, African National Congress,African-American Civil Rights Movement(1954-68),African-American culture,African-American history,Black Enterprise, African-American literature,African-American music,Afrocentric education,Afrocentrism,Afro-American religion,Angela Davis, Anglicanism,Anti-discrimination,Baptism,Baptism of Jesus,Bias,Black (Color),Black Arts Movement,Black Business Builders Club,BLACK BUSINESS GLOBAL,Black Business Works, Black Church,Black Economic Empowerment,Black Enterprise,Black Girls Rock!,Black history,Black History Month,Black is beautiful,Black Madonna, Black nationalism,Black Owned Business Network,Black Panther,Black Panther Party,Black Power,Black Women Are Beautiful,California African American Museum, Christian Church,Civil law (common law),Civil procedure,Conviction,Cop Block,Criminal justice,Criminal Justice Reform,Culture,Dallas,Elections in the United States, Employers Jesus Christ is my King,Employers Syria,End Slavery Now,Equal opportunity,Evangelicalism,Families Against Mandatory Minimums,Field of study African-american history, Fight the Power,Gospel,Grassroots Movement,Hotep,Huey P. Newton,Human rights,I Have a Dream,Jehovah's witness church, JesseJackson,Justice,Kemetism,Kwame Nkrumah,Latin,Leadership,Leadership development,Legal education,Louis Farrakhan,Lutheranism,madeas family reunion, Mahatma Gandhi,Malcolm X,Malcolm X (EI-Hajj Malik EI-Shabazz),Marcus Garvey,Marcus Garveyor Nigeria,Martial arts,Martin Luther,Martin Luther King III,Martin Luther King Jr, Martin Luther King Jr.,Martin Luther King Jr.Black (Color),Medgar Evers,Melanin,Mixed martial arts,Muhammad Ali,Multicultural Affinity African American (US), My Black is Beautiful,National Black Police Association (United Kingdom),Nelson Mandela,New Black Panther Party,Nigeria,Nigerians,Pan Africanist Congress of Azania, Pan-African colours,Pan-Africanism,Pentecostalism,PERSONAL amd HOME DEFENSE,Personal Defense,Personal Defense Network,Phaedra Parks, Police brutality in the United States,Police Brutality is a Crime,Police misconduct,Prayer,Presbyterianism,Protestantism,Racial integration,Refugees of the Syrian Civil War,Right of self-defense,RomanCatholic devotions,Rosa Parks,Safety,Say To No Racism,Self Defense Family,Self-defense,Self-defense (United States),Sister 2 Sister, Sister 2 Sistermagazine,Slavery by Another Name,Slavery in the United States,Small business owners,Social Justice Solutions,SouthAfrica,Stop Police Brutality,Stop Racism, Syria Charity,Syria-News,Syria—News,T-shirt,The African History Network,The Autobiography of Malcolm X,The Honorable Minister Louis Farrakhan,The Move, The Potter‘s House Church,The Self Defense Company,The Women's Self Defense Institute,Traditional black gospel,Understanding racial segregation in the unitedstates, Union of Huffinfton Post Writers and Bloggers,Union of Huffington Post Writers and Bloggers,United States presidential election,Violence prevention, Word of Faith,World Day of Social Justice,Worship

Group 4. Andrew Breitbart,Bible,Bill O'Reilly (political commentator),breitbart,Christianity,Christoph Hitchens,Conservatism,Conservatism in the United States,Deportation, Donald Trump,Donald Trump ,Faith,Foreign policy,Fox News Channel,Illegal immigration,Immigration law,Immigration to the United States,Jesus, Julian Assange,Laura Ingraham,Michael Savage,Michelle Malkin,Mike Huckabee,Mike Pence,Nationalidentity,Old Glory,Politics and social issues,Racism in the United States,Rand Paul, Republican Party (United States),Ron Paul,Rush Limbaugh,Sean Hannity,Stop Illegal Immigration,The Blaze,Tucker Carlson,United States Bill ofRights, United States Constitution,United States Department of Homeland Security,Welfare state,WikiLeaks

Group 5. 2nd Amendment,American Freedom,AmericanGuns,Anything About Guns,AR-15,Citizens Committee for the Right to Keep and Bear Arms (CCRKBA), Concealed carry,Concealed carry in the UnitedStates,dead hands,Employers Gun Owners of America,Firearm,From my cold,God,Gun Owners of America,Gun Rights, Gun Rights Across America,Guns,Guns & Ammo,Guns & Patriots,Guns & Weapons,Guns.com,HistoryPolitics,National Association for Gun Rights,National Rifle Association, Open carry in the United States,Open Carry Texas,Politics US politics (very conservative),Preserve our right to keep and bear arms,Protect the Second Amendment, Protecting Your Gun Rights,Right to keep and bear arms,Second Amendment Sisters,Second Amendment Supporters,Second Amendment to the United States Constitution, Students for Concealed Carry,The Second Amendment,US politics (conservative)

Group 6. Ahl al-Bayt, All PakistanMuslim League,All-american muslim culture,Allah,Ana muslim,As-salamu eleykum Or Islam Book,Field of study Arabic, Haram,History of Islam,Islam,Islam in the United States,Islamic studies,Islamicphilosophy,Islamism,Jesus Christ,Mosque,Muhammad,Muhammad al-Baqir,Muslim American Society, Muslim Brotherhood,Muslim League (Pakistan),Muslim world,Muslim Youth,Quran,Saudi Arabia,School Islam,School St. Edward's University,State of Palestine

Group 7. 100 Percent FED Up, 80Mexican,9GAG,???????????,Abu Eesa Niamatullah,African American (US),Al Jazeera,All Things Cherokee,Allah Akbr,Amenity, America the Beautiful,American Black Film Festival,American Civil War,American Civil War reenactment,American Correctional Association,American History,American Indian Movement,American IndianWars,American Patriot,American patriotism,American Revolutionary War,Americans for Prosperity, AMVETS,Anarchism,Anarcho-capitalism,Anonymous (group),Antelope Valley College,Anti-discriminationor Cop Block,Anti-fascism,Anti-racism,Anti-

Racist Action, Anti-war movement,Apple Music,Arab world,Asian American (US),Association of Chief Police Officers,Automobiles,Baltimore,Barack Obama,Be a Peacekeeper - The World Peacekeepers Movement,Being Chicano,Being Latino,Being Mexican,BEINGBLACK!!!,Bernie Sanders,Bisexuality,Black Dresses,Black Enterprise Business Report, Black Knowledge,BLACK MODELS 8. FASHION,Black Tea Patriots,Blacknews.com,BlackNewsoom,BMW,Border (1997 film),Breaking news,Broad-Based Black Economic Empowerment, BuzzFeed,California State University,Car dealership,Catholicism,Cato Institute,Cherokee,Cherokee language,Cherokee Nation,Chicano,Chicano Movement, Chicano rap,Chicks On TheRight,Children Of Inmates,Chris Kyle,Christ's Commission Fellowship,Civil and political rights,Classical liberalism,CNN,College Republicans, CollegeHumor,Color,community,Concerned Veterans for America,Confederate Flag,confederate states america,Confederate States Army,Confederate States of America, conservative daily,Conservative News Today,Conservative Republicans of Texas,Conservative Tribune,cop block,Copwatch,Corrections Corporation of America,Cracked.com, Culture of Mexico,Current events,Democratic Party (United States),Demonstration (protest),Detention (imprisonment),Detroit,Disabled American Veterans,Dixie, Dogs,donald trump,Donald Trump .,Donald Trump for President,Drudge Report,Dysfunctional Veterans,East Bay ContinuingEducation,Election, Employers NYPD,Employers Proud to be an American,Employers US. Army Reserve,Employers US. Military,Employers Veteran,Engenhariamilitar, Entertainment,Evangelism,Facebook,Facebook access (browser) Chrome,Facebook access (browser) Firefox,Facebook access (browser) Opera,Facebook access (mobile) all mobile devices,Facebook access (mobile) smartphones and tablets,Fail Blog,Fallen PoliceOfficers,Family,Family andrelationships, Far-right politics,Fascism,Federal Bureau of Prisons,Feminism,Field of study Human rights,Filming Cops,Filming Copsor Cop Block,Fitness and wellness,Flag of the United States,Flags of the Confederate States of America,Flash Games,Fly the American Flag,For America,Fox News Politics,Freckle,Free market,free music,Free software, friends bernie sanders,Funny Photo's,Funny Pics,Funny Pictures,Gary Johnson,Gay pride,Gay Rights,Google Play Music,Government,Grooveshark,Hart of Dixie,Heroes Behind The Badge,Hillary Clinton,Hispanic american culture,Hispanic and latino american culture,Hispanic culture,HispanicTV,Hispanidad,Hobbies and activi-

ties,Homeless shelter,Homosexuality,Hoodies,HuffPost Black Voices,HuffPost Politics,Human migration,Human rights activists,Human Sexuality,Humanitarian aid,Humanitarianism,Humour,I Have Decided to Follow Jesus,I Love the USA,IAm a Child of God,iFLJnny,iFunny,Illegal,imgur,Imgur,Imigra
inthe United States,Independence,Indian Country Today Media Network,Indiana,Individualism,Industry Military (Global),Industry Veterans (US),Innocence Project,Institute for Veterans and Military Families,Internet meme,Iraq and Afghanistan Veterans of America,Iraq War Veterans,Israel,ITunes,Ivanka Trump Fine Jewelry,Jesus Daily,jesus love u,Job title Coal Miner,Job title Veteran,Justice (band),Knowing Jesus,La Raza,Landscape,Landscape painting,Last.fm,Latin hip hop,Latino culture,Law enforcement,Law enforcement in the UnitedStates,Law Enforcement Life,Law Enforcement Today,Lawenforcement agency,Lesbian community,LGBT community,LGBT culture,LGBT Equality,LGBT history,LGBT rights by country,LGBT social movements,Liberal Democrats,Liberalism,Liberalismor Democratic Party (United States),Libertarian Party (United States),Libertarianism,Liberty,LOL,Love,Lowrider,M
X Memorial Foundation,Manufacturing,Maya Angelou,Media bias,Meme,Meme Center,Mercenary,Mexican american culture,Mexican American Pride,Mexican Pride,Mexico,Military,Minarchism,mother jones,Motherhood,Mud & Trucks,Multicultural Affinity Hispanic (US - All),Multicultural AffinityHispanic (US - English dominant),Mumia Abu-Jamal,Music,Muslims Are Not Terrorists,Muslims for America,MuslimStudents‘ Association,My Big Redneck Family,National Congress of American Indians,National Law Enforcement Officers Memorial,National Museum of American History,National Police Wives Association,National security,NationalResistance Movement,Native American Civil rights,Native american culture in the united states,Native American Indian Wisdom,Native American music,Native American Times,Native News Online,Native PeoplesMagazine,NBA Memes,News broadcasting,Nonviolence,Officer Down Memorial Page,Online games,Order of Merit of the PoliceForces,Our World with Black Enterprise,Pacifism,Paralyzed Veterans of America,Patient Protection and Affordable Care Act,Patriot (American Revolution),Patriot Nation,Patriotism,Peace movement,PeaceOnEarth,Philosop
is Not a Crime,Police,Police corruption,Police officer,Police Wives Unite,PoliceOne.com,Polisi militer,Political party,Politics,Politics Likely to engage with political content (conservative),Politics Likely to engage with political content (liberal),Politics US politics (lib-

eral),President of the United States,Prison Fellowship,Prison reform,Prison Versor School-to-prison pipeline,Prison Voices,Prison Wives,Prisoner,Prison-industrial complex,ProductiveMuslim,Proud to be A Muslim,Proud to be an American,Puppy,Racial equality,Rainbow flag (LGBT movement),Reason (magazine),Reddit,Redneck Nation,RedneckSocial Club,Religion,Remembrance Day,Republicans,Respect the Thin Blue line,Right Wing News,Right-wing politics,Riverside,Robert E. Lee,Rock music,Rodney King,Same-sex marriage,Same-sex marriage in the United States,Same-sex marriageor LGBT social movements,Say It Loud - I'm Black and I'm Proud,School University of NewOrleans,School University of California,Security alarm,Security guard,Shazam (service),Sheriffs in the United States,Social democracy,Social equality,Social justice,Social movement,Society,SoldadoGeneration Millennials,SoMexican,Sons of Confederate Veterans,SoundCloud,Southern Pride,Southern United States,Sports,Sports and outdoors,Spotify,St. Louis,Standing Rock Indian Reservation,State police,Stop the War Coalition,Support Law Enforcement,Support our Homeless Veterans,Support our troops,Support Our Veterans,Support Our VeteransHome Composition Veterans in home,Supporting Our Veterans,Syria,Tax,Tea Party movement,Tea Party Patriots,Ted Cruz,territory,Thank a Police officer,Thank A Soldier,The Anarchy,The Bible (TV miniseries),The Conservative,The Inmates,The Invaders,The Patriot Post,The Raw Story,The Tea Party,The Thin Blue Line,The Thin Blue Line (emblem),The Veterans Site,Thursday,Transgenderism,Trayvon Martin,Turkey,TV talkshows,U.S. Patriot Tactical,United Daughters of theConfederacy,United Nations Assistance Mission in Afghanistan,United States Armed Forces,United States Army,United States Army Reserve,United States Department of Veterans Affairs,United States Holocaust Memorial Museum,United States Senate,United StatesCongress,University of Wisconsin-Madison,US Military Veterans,US politics (moderate),Used car,USpolitics (very conservative),Veterans,Veterans (US),Veterans Advantage,Veterans benefits support,Veterans Day,Veterans For America,Veterans of ForeignWars,VeteransUnited Network,Vevo,Vietnam Veterans Against the War,Vietnam Veterans Memorial,Vietnam Veterans of America,Vietnam Veterans of America Foundation,Vietnam Vets,Vietnamveterans america,Visual perception,Walk with Jesus,White nationalism,Wives Behind the Badge,Women's fashion,World peace,Wounded Warrior Project,Yoga,Young Republicans,Zaid Shakir

A.5 Python Scripts for Data Collection, Cleaning, Analysis and Visualizations

Tokenisation Of Facebook Advertisments

```
import os.path
import numpy as np
import MySQLdb

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

keys = ['Ad␣ID',
        'Ad␣Text',
        'Ad␣Landing␣Page',
        'Ad␣Targeting␣Location',
        'Age',
        'Placements',
        'People␣Who␣Match',
        'Friends␣of␣connections',
        'Interests',
        'Excluded␣Connections',
        'Language',
        'Ad␣Impressions',
        'Ad␣Clicks',
            'Ad Spend',
            'Ad Creation Date',
            'Ad End Date']

# DB Insert Vars
adId = ""
adText = ""
adLandPage = ""
adTargetLoc = ""
age = ""
placements = ""
peopleWhoMatch = ""
friendsOfConnections = ""
interests = ""
excludedConn = ""
languages = ""
adImpressions = ""
adClicks = ""
adSpend = ""
adCreationDate = ""
adEndDate = ""

keys_alter = keys
keys_start = []
counter = 0

# 6620 is the max integer in the pdf nomenclature P(1)xxxxxxx.pdf where xxxxxx
for x in range(0, 6620):
```

```
# Reset Key-Value Pair Sorting Arrays
keys_alter = keys
keys_start = []
# DB Insert Vars; Re-Init on beggining of every loop for sanity
adId = ""
adText = ""
adLandPage = ""
adTargetLoc = ""
age = ""
placements = ""
peopleWhoMatch = ""
friendsOfConnections = ""
interests = ""
excludedConn = ""
languages = ""
adImpressions = ""
adClicks = ""
adSpend = ""
adCreationDate = ""
adEndDate = ""

# Preparing the FileName
# Use _ when using pdf2go.com and () when pdf2text.com
if (x < 10):
    fullFile = 'P_1_000000' + str(x)
elif (x < 100):
    fullFile = 'P_1_00000' + str(x)
elif (x < 1000):
    fullFile = 'P_1_0000' + str(x)
```

```
elif (x < 10000):
    fullFile = 'P_1_000' + str(x)

fullFile_inPath = "text/2017_q3_8_text/" + fullFile + ".txt"

if os.path.exists(fullFile_inPath):
    print("Starting␣for␣file␣-␣" + fullFile_inPath)
    # Temp = Contents of current file
    file = open(fullFile_inPath, "r")
    temp = file.read()

    count = 0;
    for control in keys:
        pos = temp.find(control)
        keys_start.insert(count, pos)
        count = count + 1

    sortCombined = sorted(zip(keys_start, keys_alter))

    # Length of List of the sorted KeyValue Pairs
    lenList = len(sortCombined)

    # Process the Items into a Key-Value Pair
    for i in range(0, lenList):

        # Get StarPos and EndPos for every attribute
        if sortCombined[i][0] != -1:
            lenOfAttr = len(sortCombined[i][1])
            startPos_Content = sortCombined[i][0] + lenOfAttr
```

```
          if (i + 1) < lenList:
              endPos_Content = sortCombined[i + 1][0]
          else:
              endPos_Content = len(temp) - 11
              endPos_Content = sortCombined[i][0] + lenOfAttr + 26
      else:
          startPos_Content = -1
          endPos_Content = -1

      # All Preprocess to insert into DB
      if (sortCombined[i][1] == "Ad␣ID"):
          adId = temp[startPos_Content:endPos_Content].lstrip()
      elif (sortCombined[i][1] == "Ad␣Text"):
          adText = temp[startPos_Content:endPos_Content].lstrip()
      elif (sortCombined[i][1] == "Ad␣Landing␣Page"):
          adLandPage = temp[startPos_Content:endPos_Content].lstrip().re
      elif (sortCombined[i][1] == "Ad␣Targeting␣Location"):
          adTargetLoc = temp[startPos_Content:endPos_Content].lstrip()
      elif (sortCombined[i][1] == "Age"):
          age = temp[startPos_Content:endPos_Content].lstrip()
      elif (sortCombined[i][1] == "Placements"):
          placements = temp[startPos_Content:endPos_Content].lstrip()
      elif (sortCombined[i][1] == "People␣Who␣Match"):
          peopleWhoMatch = temp[startPos_Content:endPos_Content].lstrip(
      elif (sortCombined[i][1] == "Friends␣of␣connections"):
          friendsOfConnections = temp[startPos_Content:endPos_Content].l
      elif (sortCombined[i][1] == "Interests"):
          interests = temp[startPos_Content:endPos_Content].lstrip()
      elif (sortCombined[i][1] == "Excluded␣Connections"):
```

```
                excludedConn = temp[startPos_Content:endPos_Content].lstrip()
            elif (sortCombined[i][1] == "Language"):
                languages = temp[startPos_Content:endPos_Content].lstrip()
            elif (sortCombined[i][1] == "Ad Impressions"):
                adImpressions = temp[startPos_Content:endPos_Content].lstrip()
            elif (sortCombined[i][1] == "Ad Clicks"):
                adClicks = temp[startPos_Content:endPos_Content].lstrip()
            elif (sortCombined[i][1] == "Ad Spend"):
                adSpend = temp[startPos_Content:endPos_Content].lstrip()
            elif (sortCombined[i][1] == "Ad Creation Date"):
                adCreationDate = temp[startPos_Content:endPos_Content].lstrip(
            elif (sortCombined[i][1] == "Ad End Date"):
                adEndDate = temp[startPos_Content:endPos_Content].lstrip()

        # DB Entry Here
        print("DB Entry for file - " + fullFile_inPath)
        try:
            cursor.execute(
                'INSERT into usaData (pdfConverter,year,quater,month,fileName,
                ('pdf2go', '2017', '3', '8', fullFile, adId, adText, adLandPag
                 peopleWhoMatch, friendsOfConnections, interests, excludedConn
                 adSpend, adCreationDate, adEndDate))
            db.commit()
        except Exception as e:
            print(e)

        print("End for file - " + fullFile_inPath)

print("Correct AD IDs resolved : " + str(counter))
```

***Python Scripts for Cleaning Facebook Advertisments Dataset* Tokenisation of Facebook Interest and Locations**

```
import MySQLdb

import numpy as np

stF1 = "Redactions␣Completed"
randomProcess = 0
# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT␣id,␣fileName,␣Interests␣from␣usaData"
cursor.execute(sql)
results = cursor.fetchall()

# Identifier Vars
idList = []
interestsList = []
andAlsoMatch_Bool = []
count = 0

for row in results:
    tempStr = row[2]
    tempStr = tempStr.replace("\n", "").replace(":", "").replace("Ahd␣Must␣Als
        "And␣MustAlso␣Match", "And␣Must␣Also␣Match").lstrip()

    # CASE 1
    finalInterests_BeforeOR_Case1 = []
    finalInterests_BeforeOR_Case1_COUNTER = 0
    finalInterests_AfterOR_Case1 = []
    finalInterests_AfterOR_Case1_COUNTER = 0

    # CASE 2
    finalInterests_BeforeOR_Case2 = []
    finalInterests_BeforeOR_Case2_COUNTER = 0
    finalInterests_AfterOR_Case2 = []
    finalInterests_AfterOR_Case2_COUNTER = 0

    # Get All OR Parameters
    # CASE 1
    if (tempStr.find("Must␣Also␣Match") == -1 and tempStr.find("Behaviors") ==

        splitByOr = tempStr.split("␣or")

        # If not empty Interests & Before the OR Clause
        if (len(splitByOr) > 0):
            listOfInterests_beforeOR = splitByOr[0].split(",")
            for item in listOfInterests_beforeOR:
                tempStr = item.lstrip()
                finalInterests_BeforeOR_Case1.insert(finalInterests_BeforeOR_Cas
                finalInterests_BeforeOR_Case1_COUNTER = finalInterests_BeforeOR_

        # When OR Clause is present
        if (len(splitByOr) > 1):
            listOfInterests_afterOR = splitByOr[1]
            listOfInterests_afterOR = listOfInterests_afterOR.split(",")
            for item in listOfInterests_afterOR:
                tempStr = item.replace("Interest␣expansion␣On", "").lstrip()
                if (tempStr.find(stF1) > 0):
                    tempStr = tempStr[0:tempStr.find(stF1)]

                finalInterests_AfterOR_Case1.insert(finalInterests_AfterOR_Case1_
                finalInterests_AfterOR_Case1_COUNTER = finalInterests_AfterOR_Ca



    # CASE 2
    else:
        listOfInterests_beforeANDALSOMATCH_Level1 = tempStr.split("And␣Must␣Al
        for eachItem in listOfInterests_beforeANDALSOMATCH_Level1:

            eachItem = eachItem.replace("Behaviors", ",")
            eachItem = eachItem.replace("Interests", "")
            eachItem = eachItem.replace("Interest␣expansion␣On", "")
            eachItem = eachItem.replace("|", "I")

            # NOW REDUCE THE CASE TO CASE 1
            splitByOr = eachItem.split("␣or")
            # If not empty Interests & Before the OR Clause
            if (len(splitByOr) > 0):
                listOfInterests_beforeOR = splitByOr[0].split(",")
                for item in listOfInterests_beforeOR:
                    tempStr = item.lstrip()
```

```
                    finalInterests_BeforeOR_Case2.insert(finalInterests_BeforeOR_
                    finalInterests_BeforeOR_Case2_COUNTER = finalInterests_Before

            # When OR Clause is present
            if (len(splitByOr) > 1):
                listOfInterests_afterOR = splitByOr[1]
                listOfInterests_afterOR = listOfInterests_afterOR.split(",")
                for item in listOfInterests_afterOR:
                    if (tempStr.find(stF1) > 0):
                        tempStr = tempStr[0:tempStr.find(stF1)]
                        finalInterests_AfterOR_Case2.insert(finalInterests_AfterO
                        finalInterests_AfterOR_Case2_COUNTER = finalInterests_Aft
                    else:
                        finalInterests_AfterOR_Case2.insert(finalInterests_AfterO
                        finalInterests_AfterOR_Case2_COUNTER = finalInterests_Aft

    # Aggregate all interests in one list
    tempListOfInterests = ""

    case1_BeforeOr = len(finalInterests_BeforeOR_Case1)
    case1_AfterOr = len(finalInterests_AfterOR_Case1)
    case2_BeforeOr = len(finalInterests_BeforeOR_Case2)
    case2_AfterOr = len(finalInterests_AfterOR_Case2)

    for eachItem in finalInterests_BeforeOR_Case1:
        if (tempListOfInterests == ""):
            tempListOfInterests = eachItem
        else:
            tempListOfInterests = tempListOfInterests + "," + eachItem.lstrip(
    for eachItem in finalInterests_AfterOR_Case1:
        if (tempListOfInterests == ""):
            tempListOfInterests = eachItem
        else:
            tempListOfInterests = tempListOfInterests + "," + eachItem.lstrip(

    for eachItem in finalInterests_BeforeOR_Case2:
        if (tempListOfInterests == ""):
            tempListOfInterests = eachItem
        else:
            tempListOfInterests = tempListOfInterests + "," + eachItem.lstrip(

    for eachItem in finalInterests_AfterOR_Case2:
        if (tempListOfInterests == ""):
            tempListOfInterests = eachItem
        else:
            tempListOfInterests = tempListOfInterests + "," + eachItem.lstrip(

    # Insert to KEY DS
    idList.insert(count, row[0])
    interestsList.insert(count, tempListOfInterests)
    if (case2_BeforeOr > 0 or case2_AfterOr > 0):
        andAlsoMatch_Bool.insert(count, "1")
    else:
        andAlsoMatch_Bool.insert(count, "0")

    count = count + 1
```

```
# THIS IS THE FORMULATION OF UNIQUE LIST OF INTERESTS
# Now form a list of interests
listOfInterests = []
count = 0
for eachItem in interestsList:
    tempInterestsList = eachItem.split(",")
    for eachInterest in tempInterestsList:
        if (eachInterest != "␣" or eachInterest != ""):
            listOfInterests.insert(count, eachInterest)

listOfInterests = np.unique(listOfInterests)
listOfInterests = sorted(listOfInterests)

# Insert list of Unique Interests to DB
for item in listOfInterests:
    print(item)
    try:
        cursor.execute('INSERT␣into␣interestsTable␣(interestVal)␣VALUES␣(%s)',
        db.commit()
    except Exception as e:
        print(e)

import MySQLdb
import os.path
from string import ascii_letters
from wordcloud import WordCloud, STOPWORDS
import learn as learn
import numpy as np
import matplotlib.pyplot as plt
import geonamescache
from sklearn.feature_extraction.text import CountVectorizer

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

# List of countries
countrList = [
    ('US', 'United States'),
    ('AF', 'Afghanistan'),
    ('AL', 'Albania'),
    ('DZ', 'Algeria'),
    ('AS', 'American Samoa'),
    ('AD', 'Andorra'),
    ('AO', 'Angola'),
    ('AI', 'Anguilla'),
    ('AQ', 'Antarctica'),
    ('AG', 'Antigua And Barbuda'),
    ('AR', 'Argentina'),
    ('AM', 'Armenia'),
    ('AW', 'Aruba'),
    ('AU', 'Australia'),
    ('AT', 'Austria'),
    ('AZ', 'Azerbaijan'),
    ('BS', 'Bahamas'),
    ('BH', 'Bahrain'),
    ('BD', 'Bangladesh'),
    ('BB', 'Barbados'),
    ('BY', 'Belarus'),
    ('BE', 'Belgium'),
    ('BZ', 'Belize'),
    ('BJ', 'Benin'),
    ('BM', 'Bermuda'),
    ('BT', 'Bhutan'),
    ('BO', 'Bolivia'),
    ('BA', 'Bosnia␣And␣Herzegowina'),
    ('BW', 'Botswana'),
    ('BV', 'Bouvet␣Island'),
    ('BR', 'Brazil'),
    ('BN', 'Brunei␣Darussalam'),
    ('BG', 'Bulgaria'),
    ('BF', 'Burkina␣Faso'),
    ('BI', 'Burundi'),
    ('KH', 'Cambodia'),
    ('CM', 'Cameroon'),
    ('CA', 'Canada'),
    ('CV', 'Cape␣Verde'),
    ('KY', 'Cayman␣Islands'),
    ('CF', 'Central␣African␣Rep'),
    ('TD', 'Chad'),
    ('CL', 'Chile'),
    ('CN', 'China'),
    ('CX', 'Christmas␣Island'),
    ('CC', 'Cocos␣Islands'),
    ('CO', 'Colombia'),
    ('KM', 'Comoros'),
    ('CG', 'Congo'),
    ('CK', 'Cook␣Islands'),
    ('CR', 'Costa␣Rica'),
    ('CI', 'Cote␣D`ivoire'),
    ('HR', 'Croatia'),
    ('CU', 'Cuba'),
    ('CY', 'Cyprus'),
    ('CZ', 'Czech␣Republic'),
    ('DK', 'Denmark'),
    ('DJ', 'Djibouti'),
    ('DM', 'Dominica'),
    ('DO', 'Dominican␣Republic'),
    ('TP', 'East␣Timor'),
    ('EC', 'Ecuador'),
    ('EG', 'Egypt'),
    ('SV', 'El␣Salvador'),
    ('GQ', 'Equatorial␣Guinea'),
    ('ER', 'Eritrea'),
    ('EE', 'Estonia'),
    ('ET', 'Ethiopia'),
    ('FK', 'Falkland␣Islands␣(Malvinas)'),
    ('FO', 'Faroe␣Islands'),
    ('FJ', 'Fiji'),
    ('FI', 'Finland'),
    ('FR', 'France'),
    ('GF', 'French␣Guiana'),
    ('PF', 'French␣Polynesia'),
    ('TF', 'French␣S.␣Territories'),
    ('GA', 'Gabon'),
    ('GM', 'Gambia'),
    ('DE', 'Germany'),
    ('GH', 'Ghana'),
    ('GI', 'Gibraltar'),
    ('GR', 'Greece'),
    ('GL', 'Greenland'),
    ('GD', 'Grenada'),
    ('GP', 'Guadeloupe'),
    ('GU', 'Guam'),
    ('GT', 'Guatemala'),
    ('GN', 'Guinea'),
    ('GW', 'Guinea-bissau'),
    ('GY', 'Guyana'),
    ('HT', 'Haiti'),
    ('HN', 'Honduras'),
    ('HK', 'Hong␣Kong'),
    ('HU', 'Hungary'),
    ('IS', 'Iceland'),
    ('IN', 'India'),
    ('ID', 'Indonesia'),
    ('IR', 'Iran'),
    ('IQ', 'Iraq'),
    ('IE', 'Ireland'),
    ('IL', 'Israel'),
    ('IT', 'Italy'),
    ('JM', 'Jamaica'),
    ('JP', 'Japan'),
    ('JO', 'Jordan'),
    ('KZ', 'Kazakhstan'),
    ('KE', 'Kenya'),
    ('KI', 'Kiribati'),
    ('KP', 'Korea␣(North)'),
    ('KR', 'Korea␣(South)'),
    ('KW', 'Kuwait'),
    ('KG', 'Kyrgyzstan'),
    ('LA', 'Laos'),
    ('LV', 'Latvia'),
    ('LB', 'Lebanon'),
    ('LS', 'Lesotho'),
    ('LR', 'Liberia'),
    ('LY', 'Libya'),
    ('LI', 'Liechtenstein'),
    ('LT', 'Lithuania'),
    ('LU', 'Luxembourg'),
    ('MO', 'Macau'),
    ('MK', 'Macedonia'),
    ('MG', 'Madagascar'),
    ('MW', 'Malawi'),
    ('MY', 'Malaysia'),
    ('MV', 'Maldives'),
    ('ML', 'Mali'),
    ('MT', 'Malta'),
    ('MH', 'Marshall␣Islands'),
    ('MQ', 'Martinique'),
    ('MR', 'Mauritania'),
    ('MU', 'Mauritius'),
    ('YT', 'Mayotte'),
    ('MX', 'Mexico'),
    ('FM', 'Micronesia'),
    ('MD', 'Moldova'),
    ('MC', 'Monaco'),
    ('MN', 'Mongolia'),
    ('MS', 'Montserrat'),
    ('MA', 'Morocco'),
    ('MZ', 'Mozambique'),
    ('MM', 'Myanmar'),
    ('NA', 'Namibia'),
    ('NR', 'Nauru'),
    ('NP', 'Nepal'),
    ('NL', 'Netherlands'),
    ('AN', 'Netherlands␣Antilles'),
    ('NC', 'New␣Caledonia'),
    ('NZ', 'New␣Zealand'),
    ('NI', 'Nicaragua'),
    ('NE', 'Niger'),
    ('NG', 'Nigeria'),
    ('NU', 'Niue'),
    ('NF', 'Norfolk␣Island'),
    ('MP', 'Northern␣Mariana␣Islands'),
    ('NO', 'Norway'),
    ('OM', 'Oman'),
    ('PK', 'Pakistan'),
    ('PW', 'Palau'),
    ('PA', 'Panama'),
    ('PG', 'Papua New Guinea'),
    ('PY', 'Paraguay'),
    ('PE', 'Peru'),
    ('PH', 'Philippines'),
    ('PN', 'Pitcairn'),
    ('PL', 'Poland'),
    ('PT', 'Portugal'),
    ('PR', 'Puerto Rico'),
    ('QA', 'Qatar'),
    ('RE', 'Reunion'),
    ('RO', 'Romania'),
    ('RU', 'Russian Federation'),
    ('RW', 'Rwanda'),
    ('KN', 'Saint Kitts And Nevis'),
    ('LC', 'Saint Lucia'),
    ('VC', 'St Vincent/Grenadines'),
    ('WS', 'Samoa'),
    ('SM', 'San Marino'),
    ('ST', 'Sao Tome'),
    ('SA', 'Saudi Arabia'),
    ('SN', 'Senegal'),
    ('SC', 'Seychelles'),
    ('SL', 'Sierra Leone'),
    ('SG', 'Singapore'),
    ('SK', 'Slovakia'),
    ('SI', 'Slovenia'),
    ('SB', 'Solomon Islands'),
    ('SO', 'Somalia'),
    ('ZA', 'South Africa'),
    ('ES', 'Spain'),
    ('LK', 'Sri␣Lanka'),
    ('SH', 'St.␣Helena'),
    ('PM', 'St.Pierre'),
    ('SD', 'Sudan'),
    ('SR', 'Suriname'),
    ('SZ', 'Swaziland'),
    ('SE', 'Sweden'),
    ('CH', 'Switzerland'),
    ('SY', 'Syrian␣Arab␣Republic'),
    ('TW', 'Taiwan'),
    ('TJ', 'Tajikistan'),
    ('TZ', 'Tanzania'),
    ('TH', 'Thailand'),
    ('TG', 'Togo'),
    ('TK', 'Tokelau'),
    ('TO', 'Tonga'),
    ('TT', 'Trinidad␣And␣Tobago'),
    ('TN', 'Tunisia'),
    ('TR', 'Turkey'),
    ('TM', 'Turkmenistan'),
    ('TV', 'Tuvalu'),
    ('UG', 'Uganda'),
    ('UA', 'Ukraine'),
    ('AE', 'United␣Arab␣Emirates'),
    ('UK', 'United␣Kingdom'),
    ('UY', 'Uruguay'),
    ('UZ', 'Uzbekistan'),
    ('VU', 'Vanuatu'),
    ('VA', 'Vatican␣City␣State'),
    ('VE', 'Venezuela'),
    ('VN', 'Viet␣Nam'),
    ('VG', 'Virgin␣Islands␣(British)'),
    ('VI', 'Virgin␣Islands␣(U.S.)'),
    ('EH', 'Western␣Sahara'),
    ('YE', 'Yemen'),
    ('YU', 'Yugoslavia'),
    ('ZR', 'Zaire'),
    ('ZM', 'Zambia'),
    ('ZW', 'Zimbabwe')
]

# List of USA States
usaStates = ['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colora
             'Connecticut', 'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho
             'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
             'Maine' 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
             'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
             'New␣Hampshire', 'New␣Jersey', 'New␣Mexico', 'New␣York',
             'North␣Carolina', 'North␣Dakota', 'Ohio',
             'Oklahoma', 'Oregon', 'Pennsylvania', 'Rhode␣Island',
             'South␣␣Carolina', 'South␣Dakota', 'Tennessee', 'Texas', 'Utah',
             'Vermont', 'Virginia', 'Washington', 'West␣Virginia',
             'Wisconsin', 'Wyoming']

# USA City List
gc = geonamescache.GeonamesCache()
c = gc.get_cities()
```

```
US_cities = [c[key]['name'] for key in list(c.keys())
             if c[key]['countrycode'] == 'US']

sql = "SELECT␣id,␣fileName,␣Ad_Targeting_Location␣from␣usaData"
cursor.execute(sql)
results = cursor.fetchall()

idList = []
countrListPerAd = []
satePerAd = []
cityListPerAd = []
fileList = []
count = 0

for row in results:
    tempStr = row[2]
    tempStr = tempStr.replace("/n", "")
    tempStr = tempStr.replace("-", "")
    tempStr = tempStr.replace("Living␣In:", "")
    tempStr = tempStr.replace("---", "")
    tempStr = tempStr.replace(":", "")
    if (tempStr.find("Connections␣People␣who␣like") > 0):
        tempStr = tempStr[0:tempStr.find("Connections␣People␣who␣like")]
    if (tempStr.find("Exclude␣Location") > 0):
        excludeLocation = tempStr[tempStr.find("Exclude␣Location"):len(tempStr
        tempStr = tempStr[0:tempStr.find("Exclude␣Location")]

    # Check for country
    tempCountryList = []
```

```
counter_tempCountryList = 0
for i in range(0, len(countrList)):
    if (tempStr.find(countrList[i][1]) >= 0):
        tempCountryList.insert(counter_tempCountryList, countrList[i][1])
        counter_tempCountryList = counter_tempCountryList + 1

# Check for State
tempStateList = []
counter_tempStateList = 0
for i in range(0, len(usaStates)):
    if (tempStr.find(usaStates[i]) >= 0):
        tempStateList.insert(counter_tempStateList, usaStates[i])
        counter_tempStateList = counter_tempStateList + 1

# Check for City
tempCityList = []
counter_tempCity = 0
for i in range(0, len(US_cities)):
    if (tempStr.find(US_cities[i]) >= 0):
        tempCityList.insert(counter_tempCity, US_cities[i])
        counter_tempCity = counter_tempCity + 1

print(tempCityList)

idList.insert(count, row[0])
countrListPerAd.insert(count, tempCountryList)
satePerAd.insert(count, tempStateList)
cityListPerAd.insert(count, tempCityList)
fileList.insert(count, row[1])
```

```
    count = count + 1

print("Begining DB Entry...")
for i in range(0, len(idList)):
    id_Insert = idList[i]

    # Country
    county_Insert = ""
    for j in range(0, len(countrListPerAd[i])):
        if (county_Insert == "" or i == (len(countrListPerAd[i]) - 1)):
            county_Insert = countrListPerAd[i][j]
        else:
            county_Insert = county_Insert + "," + countrListPerAd[i][j]

    # State
    state_Insert = ""
    for j in range(0, len(satePerAd[i])):
        if (state_Insert == "" or i == (len(satePerAd) - 1)):
            state_Insert = satePerAd[i][j]
        else:
            state_Insert = state_Insert + "," + satePerAd[i][j]

    # City
    city_Insert = ""
    for j in range(0, len(cityListPerAd[i])):
        if (city_Insert == "" or i == (len(cityListPerAd) - 1)):
            city_Insert = cityListPerAd[i][j]
        else:
            city_Insert = city_Insert + "," + cityListPerAd[i][j]
    # Insert to DB
    try:
        cursor.execute(
            'INSERT␣into␣locationOfAds␣(id,␣country,␣state,␣city)␣values␣(%s,␣
            (id_Insert, county_Insert, state_Insert, city_Insert))
        db.commit()



    except Exception as e:
        print(e)
```

Resolving Character Encodings

```
import MySQLdb
import os.path
from string import ascii_letters

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT␣id,␣fileName,␣Ad_Landing_Page␣FROM␣usaData␣order␣by␣id"
cursor.execute(sql)
results = cursor.fetchall()
```

```
idList = []
fileNameList = []
urlList = []
count = 0

for row in results:
    print("———Starting␣Iteration␣:␣" + str(count) + "—————")
    # Inserting id, fileName
    idList.insert(count, row[0])
    fileNameList.insert(count, row[1])

    tempStr = row[2]

    # \n
    tempStr = tempStr.replace("\n", "")

    # Empty Pages
    if (tempStr == "␣" or tempStr == "——"):
        tempStr = ""

    # Resolve last character to / only ; on scrutiny, it was found raw URLs en
    if (len(tempStr) > 0):
        lengthOfCurrenttempStr = len(tempStr)

        # Handling special chars
        if (tempStr[lengthOfCurrenttempStr - 1] == "]" or tempStr[lengthOfCurr
            tempStr = tempStr.replace("]", "/")
            tempStr = tempStr.replace("!", "/")
        # Take care of rest strings that do no end with / (chars + digits; not
        if (not tempStr.endswith("/")):
            tempStr = tempStr + "/"

    # Preprocess malformed https. On scrutiny it was found malformed strings b
    tempStr = tempStr.replace("https:H", "https://")
    tempStr = tempStr.replace("httpszll", "https://")
    tempStr = tempStr.replace("h1tps://", "https://")
    tempStr = tempStr.replace("httpzll", "https://")

    # Replace all | with l - this is a PDF conversion error
    tempStr = tempStr.replace("|", "l")

    # Variants of instagram to instagram
    if (tempStr.find("gram") >= 0 and tempStr.find("instagram") == -1):
        tempStr = tempStr.replace("ihstagram", "instagram")
        tempStr = tempStr.replace("_ihstagram", "instagram")
        tempStr = tempStr.replace("jhstagram", "instagram")
        tempStr = tempStr.replace("_instagram", "instagram")

    # Variants of facebook to facebook
    tempStr = tempStr.replace("tacebook", "facebook")

    # Preprocess all .c0m, .comN, oom, .Com, _com, _coml, .coml, .comI : Varia
    if (tempStr.find(".com") == -1):
        tempStr = tempStr.replace(".c0m", ".com")
        tempStr = tempStr.replace(".oom", ".com")
        tempStr = tempStr.replace("_oom", ".com")
        tempStr = tempStr.replace(".Com", ".com")
        tempStr = tempStr.replace("_com", ".com")
        tempStr = tempStr.replace("_coml", ".com")
    # Preprocess all .com/ : Variants found by scrutiny
    if (tempStr.find(".com/") == -1):
        tempStr = tempStr.replace(".comN", ".com")
        tempStr = tempStr.replace(".coml", ".com")
        tempStr = tempStr.replace(".comL", ".com")
        tempStr = tempStr.replace(".comI", ".com")
        tempStr = tempStr.replace(".comN", ".com")
    # Resolve com to .com
    if (tempStr.find(".com") == -1 and tempStr.find("com") >= 0):
        tempStr = tempStr.replace("com", ".com")
    # .com to .com/
    if (tempStr.find(".com/") == -1):
        tempStr = tempStr.replace(".com", ".com/")

    # Resolve all www, variants found by scrutiny
    if (tempStr.find("www.") == -1):
        tempStr = tempStr.replace("www", "www.")
        tempStr = tempStr.replace("www_", "www.")
        tempStr = tempStr.replace("V\ANW", "www.")
        tempStr = tempStr.replace("www._", "www.")

    # Common preprocessing error of WilliamsandKalvin
    if (tempStr.find("andKalvin") > 1):
        tempStr = "https://www.facebook.com/WilliamsandKalvin/"

    # Common error with 788980617892144 to https://www.facebook.com/WilliamsK
    if (tempStr.find("788980617892144") > 1):
        tempStr = "https://www.facebook.com/WilliamsKalvin-788980617892144/"

    # Identify and replace all --- with - for char set consistency (the first
    tempStr = tempStr.replace("---", "-")

    # Preprocess "AdTargetingCustomLocation"
    if (tempStr.find("AdTargetingCustomAudience") > 0):
        pos = tempStr.find("AdTargetingCustomAudience")
        tempStr = tempStr[0:pos]

    # Preprocess 'l to "", I to ""
    tempStr = tempStr.replace("'l", "")
    tempStr = tempStr.replace("I", "")
    tempStr = tempStr.replace("'/","")
    tempStr = tempStr.replace("'f","")
    # Preprocess '? to ?
    tempStr = tempStr.replace("'?", "")

    # Preprocess N0thing to nothing
    tempStr = tempStr.replace("N0thing", "Nothing")

    # levents,eventsl to /events/
    tempStr = tempStr.replace("levents", "/events")
    tempStr = tempStr.replace("eventsl", "events/")

    #Replace .com// with .com
    tempStr = tempStr.replace(".com//", ".com/")

    # Force changes :
```

```
    # a) vww.facebook.com/Black-Matters-1579673598947501/ => https://ww.facebo
    # b) Noke-Blacks-294234600956431/ => https://www.facebook.com/Voke-Blacks-
    if (tempStr.find("vww.facebook.com/Black-Matters-1579673598947501/") > 0):
        tempStr = "https://www.facebook.com/Black-Matters-1579673598947501/"
    if (tempStr.find("Noke-Blacks-294234600956431/") > 0):
        tempStr = "https://www.facebook.com/Voke-Blacks-294234600956431/"

    # Insert to URL List
    urlList.insert(count, tempStr)

    print("----Ending Iteration : " + str(count) + "--------")
    count = count + 1

unique = set(urlList)
print("UNIQUE_SIZE : " + str(len(unique)))

print("Begin DB Operations..")

print("Length of idList : " + str(len(idList)))
print("Length of urlList : " + str(len(urlList)))
print("Length of fileName : " + str(len(fileNameList)))

for i in range(0, len(urlList)):
    try:
        cursor.execute("""UPDATE usaData SET preprocessed_AdLandingPage=%s WHE
        db.commit()
        print("Updated id : "+str(idList[i])+" with : "+str(urlList[i]))
    except Exception as e:
        print(e)
```

```
import MySQLdb
import fuzzywuzzy
from fuzzywuzzy import fuzz

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT interestVal FROM interestsTable"
cursor.execute(sql)
results = cursor.fetchall()

listInterests = []
seenList = []
count = 0
for item in results:
    listInterests.insert(count, item[0])
    seenList.insert(count, 0)
    count = count + 1

count = 0
countMain = 0
for row in listInterests:
    if (seenList[countMain] == 0):
        currentInterest = row
        print("Doing for : " + row)
        print("==========================================================")
        listOfSimilarity = []
        count = 0
        for allInterests in results:
            x = fuzz.ratio(currentInterest, allInterests[0])
            if (x > 90):
                listOfSimilarity.insert(count, allInterests[0])
                count = count + 1

        print(listOfSimilarity)

        # Replacement Logic : Update Seen List + Update DB
        var = input("Enter any non-int string to continue OR String to replace

        if (var != ""):
            for i in range(0, len(listOfSimilarity)):

                # UPDATE SEEN LIST
                for n, j in enumerate(listInterests):
                    if (j == listOfSimilarity[i]):
                        seenList[n] = 1

                # Updating the comma separated interests table
                sqlFetchAllInterests = "SELECT id, interests from locationOfAds
                cursor.execute(sqlFetchAllInterests)
                resultsAllInterests = cursor.fetchall()
                for all in resultsAllInterests:
                    if (all[1].find(listOfSimilarity[i]) >= 0):
                        tempStr = all[1]
                        tempStr = tempStr.replace(listOfSimilarity[i], var)
                        tempID = str(all[0])
                        try:
                            cursor.execute(
                                """UPDATE locationOfAds SET interests=%s WHERE
                                (tempStr, tempID))
                            db.commit()
                        except Exception as e:
                            print(e)

                # Updating the list of interests table
                sqlFetchAllUniqueInterests = "SELECT␣id,␣interestVal␣from␣inte
                cursor.execute(sqlFetchAllUniqueInterests)
                resultsAllUniqueInterests = cursor.fetchall()
                for all in resultsAllUniqueInterests:
                    if (all[1].find(listOfSimilarity[i]) >= 0):
                        tempStr = all[1]
                        tempStr.replace(listOfSimilarity[i], var)
                        tempID = str(all[0])
                        try:
                            cursor.execute(
                                """UPDATE interestsTable SET interestVal=%s WH
                                (var, tempID))
                            db.commit()
                        except Exception as e:
                            print(e)

        print("===============================================")
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
```

```
        print("\n")

    countMain = countMain + 1
```

Assignment of Relevant Datatypes from String in Text Files

```
import MySQLdb
import os.path
from string import ascii_letters

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT␣id,␣fileName,␣Ad_Clicks␣FROM␣usaData␣order␣by␣id"
cursor.execute(sql)
results = cursor.fetchall()

idList = []
fileList = []
processedClicks = []
count = 0

for row in results:
    temp = row[2].lstrip().replace("\n", "").replace(",", "").replace("␣", "")
    try:
        temp = int(temp)
        print(temp)
    except Exception as e:
        print("Error, please do manual input : " + temp + " id : " + str(row[0
        var = input("Please enter click count : ")
        temp = int(var)

    processedClicks.insert(count, temp)
    idList.insert(count, row[0])
    fileList.insert(count, row[1])
    count = count + 1

print("Begin DB Operations..")
for i in range(0, len(idList)):



    try:
        cursor.execute("""UPDATE usaData SET preprocessed_AdClicks0=%s WHERE i
        db.commit()
        print("Impression : " + str(processedClicks[i]) + " with ID : " + str(
    except Exception as e:
        print(e)

import MySQLdb
import os.path
from string import ascii_letters

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT id, fileName, Ad_Impressions FROM usaData order by id"
cursor.execute(sql)
results = cursor.fetchall()

idList = []
fileList = []
processedImpression = []
count = 0

for row in results:
    temp = row[2].lstrip().replace("\n", "").replace(",", "").replace(" ", "")
    try:
        temp = int(temp)
        print(temp)
    except Exception as e:
        print("Error, please do manual input : " + temp + " id : " + str(row[0
        var = input("Please enter impression count : ")
        temp = int(var)

    processedImpression.insert(count, temp)
    idList.insert(count, row[0])
    fileList.insert(count, row[1])
    count = count + 1
```

```
print("Begin DB Operations..")
for i in range(0, len(idList)):



    try:
        cursor.execute("""UPDATE usaData SET preprocessed_AdImpressions=%s WHE
        db.commit()
        print("Impression : " + str(processedImpression[i]) + " with ID : " +
    except Exception as e:
        print(e)

import MySQLdb
import os.path
from string import ascii_letters

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT id, fileName, Ad_End_Date FROM usaData order by id"
cursor.execute(sql)
results = cursor.fetchall()

idList = []
```

```
fileNameList = []
dateList = []
timeList = []
am_pmList = []
timezoneList = []
count = 0

for row in results:
    tempStr = row[2].replace("␣", "")
    if(len(tempStr)>0):
        date = tempStr[0:8]
        time = tempStr[8:16].replace("-",":")
        am_pm = tempStr[16:18]
        timezone = tempStr[18:21]
    else:
        date = ""
        time = ""
        am_pm = ""
        timezone = ""



    #Preprocess all 2,5 position that are not :
    if (len(time)>0 and time[2] == "2" and time[5] == "2"):
        time = time[0:2] + ":" +time[3:5] + ":" +time[6:8]
    elif (len(time)>0 and time[2] == "1" and time[5] == "1"):
        time = time[0:2] + ":" + time[3:5] + ":" +time[6:8]
    elif (len(time)>0 and time[2] == "z" and time[5] == "z"):
        time = time[0:2] + ":" + time[3:5] + ":" +time[6:8]
    idList.insert(count, row[0])
    fileNameList.insert(count, row[1])
    dateList.insert(count, date)
    timeList.insert(count, time)
    am_pmList.insert(count, am_pm)
    timezoneList.insert(count, timezone)

    count = count + 1

print("Begin DB Operations..")
for i in range(0, len(idList)):



    try:
        cursor.execute("""UPDATE usaData SET preprocess_endDate=%s, preprocess_
        db.commit()
        print("End Date : " + str(dateList[i]) + " with ID : " + str(idList[i]
    except Exception as e:
        print(e)

import MySQLdb
import os.path
from string import ascii_letters

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT␣id,␣fileName,␣Ad_Creation_Date␣FROM␣usaData␣order␣by␣id"
cursor.execute(sql)
results = cursor.fetchall()

idList = []
fileNameList = []
dateList = []
timeList = []
am_pmList = []
timezoneList = []
count = 0

for row in results:
    tempStr = row[2].replace("␣", "")
    date = tempStr[0:8]
    time = tempStr[8:16]
    am_pm = tempStr[16:18]
    timezone = tempStr[18:21]

    idList.insert(count, row[0])
    fileNameList.insert(count, row[1])
    dateList.insert(count, date)
    timeList.insert(count, time)
    am_pmList.insert(count, am_pm)
    timezoneList.insert(count, timezone)

    count = count + 1

print("Begin␣DB␣Operations..")
for i in range(0, len(idList)):



    try:
        cursor.execute("""UPDATE usaData SET preprocess_startDate=%s, preproce
        db.commit()
        print("Start␣Date␣:␣" + str(dateList[i]) + "␣with␣ID␣:␣" + str(idList[
    except Exception as e:
        print(e)
```

Python Scripts for Collecting Google Trends Data **Collection Google Trends Data for Entire USA**

```
import MySQLdb
import pandas as pd
import pytrends
from pytrends.request import TrendReq
import pandas as pd
from datetime import date, time
from pytrends.request import TrendReq
import time

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT␣*␣FROM␣interestsTable␣WHERE␣interestVal!=''␣and␣interestVal!='20
cursor.execute(sql)
results = cursor.fetchall()

StartDate = "2015-01-01"
EndDate = "2018-01-01"

pytrend = TrendReq(hl='en-US')

count = 0
for each in results:
    curInterest = each
    if (count >= 90):
        time.sleep(90)
        print("sleeping..")
        count = 0

    print("Request␣Counts␣:␣" + str(count))
    count = count + 1
    print(curInterest)

    regionQuery = "US"
    pytrend.build_payload(kw_list=[curInterest], timeframe=StartDate + "␣" + E
                          geo=regionQuery)
    interest_over_time_df = pytrend.interest_over_time()
    print("==============")
    # Dump the data
    fileName = "dataDump_entireUSA/" + curInterest + ".csv"
    interest_over_time_df.to_csv(fileName, index=False, sep=',', encoding='utf
```

Collection Google Trends Data at a State Level

```
import MySQLdb
import pandas as pd
import pytrends
from pytrends.request import TrendReq
import pandas as pd
from datetime import date, time
from pytrends.request import TrendReq
import time

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

# List of State Abbreviations
stateAbbers = []
stateName = []
count = 0
stateLists = pd.read_csv("states.csv", delimiter=",")
for index, row in stateLists.iterrows():
    stateAbbers.insert(count, row[1])
    stateName.insert(count, row[0])
    count = count + 1

sql = "SELECT␣*␣FROM␣interestsTable␣WHERE␣interestVal!=''"
cursor.execute(sql)
results = cursor.fetchall()

StartDate = "2015-01-01"
EndDate = "2018-01-01"

pytrend = TrendReq(hl='en-US')
count = 0
for each in results:
    curInterest = each
    for i in range(0, len(stateAbbers)):
        currentRegion = stateAbbers[i]
        sql = "SELECT␣COUNT(*)␣FROM␣locationOfAds␣WHERE␣state␣LIKE('%" + stateN
            i] + "%')␣AND␣interests␣LIKE('%" + curInterest + "%')"
        cursor.execute(sql)
        res = cursor.fetchall()

        if (res[0][0] > 0):
            if (count >= 90):
                time.sleep(90)
                print("sleeping..")
                count = 0
            print("Request␣Counts␣:␣" + str(count))
            count = count + 1
            print(curInterest)
            print(currentRegion)

            regionQuery = "US" + "-" + currentRegion
            pytrend.build_payload(kw_list=[curInterest], timeframe=StartDate +
                                  geo=regionQuery)
            interest_over_time_df = pytrend.interest_over_time()
            print("=============")
            # Dump the data
            fileName = "dataDump/" + curInterest + "_" + stateAbbers[i] + "_"
            interest_over_time_df.to_csv(fileName, index=False, sep=',', encod
```

Python Scripts for Crawling and Cleaning Twitter Mentions Dataset **Crawling all Mentions for each Twitter Troll Handle**

```
import twitterpastcrawler

# Open File of all Twitter Handels
fullFile_inPath = "../cleanData/cleanedHandels.txt"
file = open(fullFile_inPath, "r")

twitter_handels = file.read()

# Fetch Data for Each Twitter Handel
curStr = ""
for each in twitter_handels:
    if (each != "\n"):
        curStr = curStr + each
    else:
        curStr = curStr.lstrip()
        print("==============================")
        print("Getting␣Data␣for␣:␣" + curStr)
        queryStr = "@" + curStr
        saveLoc = "../fetchedData/" + curStr + ".csv"

        try:
            crawler = twitterpastcrawler.TwitterCrawler(
                query=queryStr,
                output_file=saveLoc
            )
            crawler.crawl()
        except Exception as e:
            print(e)

        curStr = ""
```

Fetching Twitter user id Information and Botometer Scores

```
#IRA Data analysis
#basic count
#all users info
#high mentioners' count and their bot score
setwd("I:/twitter␣KCL/otherStuff/IRATwitter/")
#My code utility functions
source("I:/twitter␣KCL/getUtilities.R")

#user IDs
iraId <- fileRead("rawDataIRA.csv")
iraId <- fileRead("rawDataIRA2.csv")
```

```
#find IDs counts
count <- sortedTable(iraId$userID)

sum(count[which(as.numeric(count)>0)])

(7000)/360/24

plotCDF(as.numeric(count),log = T,"IRA Interaction Count")

count[1:10]
#user info
length(count)
user=as.numeric(names(count))
#run and get res tweetUserProfile()
#gives all users profile details
res <- tweetUserProfile(user)
res <- cbind(freq=as.numeric(count),res)
#delete with threshold less than value
iraUserProfile <- deleteNAVec(res,df = T,3)
countNA(iraUserProfile$screen_name)

write.csv(iraUserProfile,"iraUserProfile2.csv")

#second IRA file
newira <- fileRead("linkedUserID.csv")
newira <- newira[which(newira$score==-2),]
write.csv(newira,"iraLinkedUserID.csv")
```

```
clickInf <- fileRead("clickInfluence.csv")
plotCDF(clickInf$clickInfluence ,xLab = "#Population",grid = T)

impressionInf <- fileRead("impressionInfluence.csv")

plotCDFAdd(cdf2=impressionInf$impressionInfluence ,legn=c("Clicks","Impression"



#Function for tweeter profiles-
####User profile match, work both with user ids a and screenNames####
tweetUserProfile <- function(user=,fname,
                                  reset=F,useDump=F){
  #look user profile , unique id/names, then merge back
  #pass id from json else screenname fromm csv data
  #user=c("pushkalagarwa","pushkalagarwa")
  if(reset==T)
    write.csv(lookup_users("pushkalagarwa"),"I:/twitter_KCL/otherStuff/BigUserI
  #no effect on ids
  #check any user missing
  id <- unique(user)
  if(useDump==T){
    lookNameData <- fileRead("I:/twitter_KCL/otherStuff/BigUserDump.csv")
    if(typeof(user)=="double"|typeof(user)=="integer"){
      lookNameData$user_id <-as.numeric(lookNameData$user_id)
      ind <- id%in%unique(lookNameData$user_id)
    }else{
      ind <- id%in%unique(lookNameData$screen_name)
      lookNameData$screen_name <-tolower(lookNameData$screen_name)
      user <- tolower(user)
    }
  }else{
    ind <- rep(F,length(id))
    lookNameData <- c()
  }

  cat("Missing:",length(which(ind==F)),"\n")
  if(length(which(ind==F))>0){
    #remaing fetch
    id <- id[!ind]
    #seq for i to j with by=50K, last variable
    #make sure rate limit is ok,so use small chunks
    ind <- c(seq(1,length(id),50000),length(id))
    i <- 1
    #stores profile data in chucks of 50K
    tempName <- c()
    #make efficient for sync
    for(i in 1:(length(ind)-1)){
      time <- Sys.time()
      cat(ind[i]," ",ind[i+1]-1,"\n")
      while(length(lookup_users(user[1]))==0){
        cat("Sleeping")
        Sys.sleep(120)
      }
      tempName <- rbind(tempName,lookup_users(id[ind[i]:(ind[i+1]-1)]))
      sleepTime <- as.numeric(abs(Sys.time()-(time)-15))*60
      cat("Sleeping for-",sleepTime)
      Sys.sleep(sleepTime)
    }
    #if all null then skip else update big file
    if(nrow(tempName)>0){
      #name data is tweets data with user profile
      #careful with . eof description and lower cases
      if(typeof(user)=="double"|typeof(user)=="integer"){
        lookMissingData<- tempName[match(id,tempName$user_id),]
      }else{
        lookMissingData<- tempName[match(tolower(id),tolower(tempName$screen_n
      }
      cat("\nNA,␣Missing-",countNA(lookMissingData[,1]))
      lookNameData <- rbind(lookNameData,lookMissingData)
      #remove repeated
      #lookNameData <- lookNameData[fileUniqueInd(lookNameData$user_id),]
      #cat("\nBig File now",nrow(lookNameData))
      #write.csv(lookNameData,"I:/twitter KCL/otherStuff/BigUserDump.csv",row.
      #lookNameData <- rbind(lookNameData[ind,-1],lookMissingData)
    }
  }

  #carefull with lower
  if(typeof(user)=="double"|typeof(user)=="integer")
  {
    res <- lookNameData[match(user,as.numeric(lookNameData$user_id)),]
  }else{
    res <- lookNameData[match(tolower(user),tolower(lookNameData$screen_name))
  }
  #res <- res[fileUniqueInd(res$user_id),]
  cat("\nWarning:␣NA␣found-",countNA(res[,1]))
  res
}


#botometer code

#install.packages("devtools")
library(devtools)
#install_github("marsha5813/botcheck")
library(botcheck)

# Load dependencies
library(httr)
library(xml2)
library(RJSONIO)


#You'll also need to create and register a Twitter app. Save your keys by repl

#Call the botcheck() function to get the probability that a Twitter handle is

sortedTable <- function(val){
  sort(table(val),decreasing = T)
}
cat("Config␣Done,␣reading␣file\n")
#check for deleted and protected in advance
# r <- read.csv("MentionUsers.csv",stringsAsFactor=F)
# sn <- names(sortedTable(r$snName))
r <- read.csv("iraUserProfile.csv",stringsAsFactor=F)
```

```
sn <- r$screen_name
#sn <- names(sortedTable(r$screen_name))

rm(r)

#read all, remove fetched next fetch
snScore <- read.csv("iraBotCheck.csv",stringsAsFactors = F)
#snScore <- read.csv("10TopBotCheck.csv",stringsAsFactors = F)
#sn <- snScore$user
sn <- sn[-which(sn%in%snScore$user==T)]
cat(format(Sys.time(), "%a␣%b␣%d␣%X␣%Y"),"\nsn␣and␣snScore␣now-", length(sn),"
i <- 1
check <- c()
#now for later foreach
#careful with nlp content function()
# do for 600 per hour, if not deleted accounts
if(length(sn)<1)
  return
for(i in 1:length(sn[1:600])){
    b <- 0
    x <- sn[i]
    b <- try(botcheck(x))
    if(is.null(b)){
      while(is.null(botcheck(sn[1]))){
        cat("Sleeping...........\n")
        Sys.sleep(60)
      }
      b <- try(botcheck(x))
      if(is.null(b)){
        b=-1
      }
    }
        cat(i,x,b,"\n")
    check=rbind(check,data.frame(user=x,prob=b,stringsAsFactors = F))
}
snScore <- snScore[,-1]
check <- rbind(snScore,check)
#write.csv(check,"dmBotCheck.csv")
write.csv(check,"iraBotCheck.csv")

cat(format(Sys.time(), "%a %b %d %X %Y"),"Exited\n")
```

Generation of PDF for Frequency of Tweets of Each Unique User

```
import MySQLdb
import matplotlib.pyplot as plt
import numpy as np
import math

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usaTwitterData")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

print("fetching..")
sql = "SELECT userID,COUNT(*) as count FROM rawData GROUP BY userID ORDER BY c
cursor.execute(sql)
```

```
results = cursor.fetchall()
print("fetched..")
freqList = []
count = 0
print("prpeing␣list..")



tempList1= []
tempCnt = 0

for each in results:
    if(each[1]>=10):
        tempList1.insert(tempCnt, each[1])
        tempCnt+=1
    print(each[1])
    freqList.insert(count, math.log(each[1]))
    count = count + 1

print("Length␣:␣" + str(len(freqList)))

print("doing␣cdf..")
num_bins = 450
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(freqList) / len(freqList)
# Plot the Histogram
counts, bin_edges = np.histogram(freqList, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Frequency␣of␣Tweets␣by␣Each␣Unique␣User␣(in␣log␣base␣10)")
plt.title("CDF␣for␣frequency␣of␣Tweets␣by␣Each␣Unique␣User")
plt.grid(alpha=0.8)
plt.show()



print(len(tempList1))
sum = 0
for each in tempList1:
    sum = sum + each

print(sum)
```

Clicks Per Impressions

```
import MySQLdb
import matplotlib.pyplot as plt

# DB CONN
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

adsByQuater = []
count = 0
for yearCount in range(2015, 2018):
    for quaterCount in range(1, 5):
        sql = "SELECT␣usaData.id,usaData.preprocessed_AdImpressions,usaData.pr
            yearCount) + "␣AND␣quater=" + str(
            quaterCount) + "␣ORDER␣BY␣ID"

        cursor.execute(sql)
        results = cursor.fetchall()
        clickCount = 0
        for row in results:
            if (row[1] == 0):
                clickCount = clickCount + 0
            else:
                clickCount = clickCount + (row[2]/row[1])
        adsByQuater.insert(count, clickCount)
        count = count + 1

labels = ["2015/1","2015/2","2015/3","2015/4","2016/1","2016/2","2016/3","2016

print("~~~~~~~")
x = []
count = 0
for i in range(0, 12):
    x.insert(count, i)
    count = count + 1

fig, ax = plt.subplots()

for each in adsByQuater:
    print(each)
```

```
plt.plot(adsByQuater, color="green")
plt.ylabel('Clicks/Impression')
plt.xlabel('Quaterly␣Data')
plt.xticks(x, labels, rotation='vertical', fontsize=8)
plt.grid()
plt.show()
```

CPC/CTR Percetage of US Population exposed to IRA Adverts across USA

```
import MySQLdb
import csv
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
import pandas as pd

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

# Get Census Data
f = open("statePopulation.csv", 'r')
stateList = []
populationCount = []
count = 0
```

```
with f:
    reader = csv.reader(f)
    for row in reader:
        stateList.insert(count, row[1].lstrip().rstrip())
        populationCount.insert(count, int(row[2].replace(",", "")))
        count = count + 1

# Get all State and Ads
sql = "SELECT␣locationOfAds.country,␣locationOfAds.state,␣usaData.preprocessed␣
cursor.execute(sql)
results = cursor.fetchall()

stateCombinations = []
count = 0

impressionList = []
clickList = []
count = 0
for each in results:
    # Get Population Array
    if (each[1] != ""):
        currentStateList = each[1].split(",")
        totalPop = 0
        for eachState in currentStateList:
            indexOfState = stateList.index(eachState)
            populationOfState = populationCount[indexOfState]
            totalPop = totalPop + populationOfState
        impressionList.insert(count, (each[2] / totalPop) * 100)
        clickList.insert(count, (each[3] / totalPop) * 100)
```

```
            count = count + 1
        else:
            totalPop = 323405935
            impressionList.insert(count, (each[2] / totalPop))
            clickList.insert(count, (each[3] / totalPop))
            count = count + 1

print("Impression")
print(str(np.sum(impressionList)))
print(str(np.sum(impressionList) / 100))
print("Clicks")
print(str(np.sum(clickList)))
print(str(np.sum(clickList) / 100))
# # For Impressions
# num_bins = 10
# fig, ax = plt.subplots()
# # Normalising the PDF
# weights = np.ones_like(impressionList) / len(impressionList)
# # Plot the Histogram
# counts, bin_edges = np.histogram(impressionList, bins=num_bins, weights=weig
# cdf = np.cumsum(counts)
# plt.plot(bin_edges[1:], cdf)
# plt.ylabel("CDF")
# plt.xlabel("Population of USA in Percentage")
# plt.title("CDF for Percentage of USA Population Who Viewed on IRA Published
# plt.grid(alpha=0.8)
# #
# # # For Clicks
# num_bins = 10
# fig, ax = plt.subplots()
## Normalising the PDF
# weights = np.ones_like(clickList) / len(clickList)
## Plot the Histogram
# counts, bin_edges = np.histogram(clickList, bins=num_bins, weights=weights)
# cdf = np.cumsum(counts)
# plt.plot(bin_edges[1:], cdf)
# plt.ylabel("CDF")
# plt.xlabel("Population of USA in Percentage")
# plt.title("CDF for Percentage of USA Population Who Clicked on IRA Published
# plt.grid(alpha=0.8)
# plt.show()
```

Percetage of US Population exposed to IRA Adverts with Respect To Swing States

```
import MySQLdb
import csv
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
import pandas as pd


try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)
```

```
# Get Census Data
f = open("statePopulation.csv", 'r')
stateList = []
populationCount = []
count = 0
with f:
    reader = csv.reader(f)
    for row in reader:
        stateList.insert(count, row[1].lstrip().rstrip())
        populationCount.insert(count, int(row[2].replace(",", "")))
        count = count + 1

# Get all State and Ads
sql = "SELECT␣locationOfAds.state,␣usaData.preprocessed_AdImpressions,␣usaData
cursor.execute(sql)
results = cursor.fetchall()

stateCombinations = []
count = 0

impressionList = []
clickList = []
count = 0
for each in results:
    # Get Population Array
    if (each[0] != ""):
        currentStateList = each[0].split(",")
        totalPop = 0
        for eachState in currentStateList:
            indexOfState = stateList.index(eachState)
            populationOfState = populationCount[indexOfState]
            totalPop = totalPop + populationOfState
        impressionList.insert(count, (each[1] / totalPop) * 100)
        clickList.insert(count, (each[2] / totalPop) * 100)
        count = count + 1

print(str(np.sum(impressionList) / 100))
print(str(np.sum(clickList) / 100))

# print("Impression")
# totalIMpressions = 0
# for each in impressionList:
#     totalIMpressions = totalIMpressions + each
# print(str(totalIMpressions/len(impressionList)*100))
#
# print("Clicks")
# totalIMpressions = 0
# for each in clickList:
#     totalIMpressions = totalIMpressions + each
# print(str(totalIMpressions/len(clickList)*100))

# For Impressions
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(impressionList) / len(impressionList)
# Plot the Histogram
counts, bin_edges = np.histogram(impressionList, bins=num_bins, weights=weight
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Population␣of␣USA␣in␣Percentage")
plt.title("CDF␣for␣Percentage␣of␣USA␣Population␣Who␣Viewed␣on␣IRA␣Published␣A
plt.grid(alpha=0.8)
#
# # For Clicks
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(clickList) / len(clickList)
# Plot the Histogram
counts, bin_edges = np.histogram(clickList, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Population␣of␣USA␣in␣Percentage")
plt.title("CDF␣for␣Percentage␣of␣USA␣Population␣Who␣Clicked␣on␣IRA␣Published␣
plt.grid(alpha=0.8)
plt.show()

import MySQLdb
import os.path
from string import ascii_letters


import pylab
from wordcloud import WordCloud, STOPWORDS
import learn as learn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

ctr_quater = []
cpc_quater = []
cpm_quater = []
count = 0
# ctr_quater.insert(count, 0)
# count = count + 1
for yearCount in range(2015, 2018):

    for quaterCount in range(1, 5):

        sql = "SELECT id,preprocessed_AdImpressions,preprocessed_AdClicks,prep
            yearCount) + " AND quater=" + str(
            quaterCount) + " ORDER BY ID"
        cursor.execute(sql)
        results = cursor.fetchall()

        # Cost Per Click
        cpc = 0
        # Click Through Ratio
        ctr = 0
        for row in results:
            # CPC
            if (row[3] == 0):
                spend = 0
            else:
                if (row[4] == "USD"):
                    spend = row[3] * 61
                else:
                    spend = row[3]
            if (spend != 0):
                cpc = cpc + float(row[2]) / float(spend)
            else:
                cpc = cpc + 0
            # CTR
            if (row[2] == 0):
                ctr = ctr + 0
            else:
                ctr = ctr + (row[2] / row[1])

        ctr_quater.insert(count, ((ctr / 120)))
        cpc_quater.insert(count, (cpc / 120))
        count = count + 1

labels = []
# labels.insert(0, "-")
count = 0
```

```
for i in range(1, 13):
    if (i < 5):
        labels.insert(count, ("2015/" + str(i)))
    elif (i < 9 and i >= 5):
        labels.insert(count, ("2016/" + str(i - 4)))
    else:
        labels.insert(count, ("2017/" + str(i - 8)))

    count = count + 1

x = []
count = 0
for i in range(0, 13):
    x.insert(count, i)
    count = count + 1

cpc_ctrList = []
count = 0
for i in range(0, len(ctr_quater)):
    if (ctr_quater[i] == 0):
        print(str(ctr_quater[i]) + "␣==>␣" + str(cpc_quater[i]) + "==>" + str(
        cpc_ctrList.insert(count, 0)
    else:
        print(str(ctr_quater[i]) + "␣==>␣" + str(cpc_quater[i]) + "==>" + str(
        cpc_ctrList.insert(count, cpc_quater[i] / ctr_quater[i])

    count = count + 1

fig, ax = plt.subplots()
```

```
print(ctr_quater)
plt.plot(ctr_quater, color="green", label="CTR in %")
plt.plot(cpc_quater, color="red", label="CPC")
plt.xlabel('Time')
plt.xticks(x, labels, rotation='vertical', fontsize=8)
ax.legend(loc='best')
plt.show()

fig, ax = plt.subplots()
print(ctr_quater)
plt.plot(cpc_ctrList, color="blue", label="CPC/CTR")
plt.ylabel("CPC/CTR")
plt.xlabel('Time With Respect To Quater Of Each Year')
plt.xticks(x, labels, rotation='vertical', fontsize=8)
ax.legend(loc='best')
plt.grid()
plt.show()
```

Frequency Of Ads By Location

```
import MySQLdb
import os.path
from string import ascii_letters

import geonamescache
from wordcloud import WordCloud, STOPWORDS
import learn as learn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
```

```
# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT * FROM stateList WHERE advertCount>=10 ORDER by advertCount DESC
cursor.execute(sql)
results = cursor.fetchall()

noAds = []
location = []
count = 0

for each in results:
    noAds.insert(count, each[3])
    location.insert(count, each[1])
    count = count + 1

# Plot the graph
y_pos = np.arange(len(location))
plt.bar(y_pos, noAds, align='center', alpha=0.5)
plt.xticks(y_pos, location)
plt.xticks(rotation=90)
plt.ylabel('No. of Ads')
plt.title('USA States')
plt.show()
```

Word Cloud

```
import MySQLdb
import os.path
from string import ascii_letters
from wordcloud import WordCloud, STOPWORDS
import learn as learn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer

# DB Conn
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "SELECT␣Ad_Text␣FROM␣usaData␣order␣by␣id"
cursor.execute(sql)
results = cursor.fetchall()

worldeText = ""
count = 0
for row in results:
    worldeText = worldeText + row[0]
    print(count)
    count = count + 1

print(worldeText)




# Generate a word cloud image
wordcloud = WordCloud().generate(worldeText)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")

# lower max_font_size
wordcloud = WordCloud(max_font_size=40).generate(worldeText)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

PDf of P(State|Interest)

```
import MySQLdb
import numpy as np
import scipy
from scipy.stats import rankdata, norm, mode, binned_statistic
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import simplejson

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "usa_securityCouncil")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sqlStates = "SELECT␣*␣FROM␣stateList"
cursor.execute(sqlStates)
allStates = cursor.fetchall()

sql = "SELECT␣*␣FROM␣conditionalAdvertsInterests␣WHERE␣p_loc_interests>0.0"
cursor.execute(sql)
results = cursor.fetchall()



probList = []
count = 0
for all in results:
    probList.insert(count, all[3])
    count = count + 1

num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(probList) / len(probList)
```

```
# Plot the Histogram
n, bins, patches = ax.hist(probList, num_bins, weights=weights, alpha=0.2, col
plt.xlabel("P␣(State␣|␣Interests)")
plt.ylabel("PDF")
plt.title("PDF␣for␣P␣(State␣|␣Interests)")
plt.grid(alpha=0.8)
plt.show()
```

CDF Generation for Audience Reached based on US Population

```
import MySQLdb
import csv
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
import pandas as pd

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

# Get Census Data
f = open("statePopulation.csv", 'r')
stateList = []
populationCount = []
count = 0
with f:
    reader = csv.reader(f)
    for row in reader:
        stateList.insert(count, row[1].lstrip().rstrip())
        populationCount.insert(count, int(row[2].replace(",", "")))
        count = count + 1

# Get all State and Ads
sql = "SELECT␣locationOfAds.state,␣usaData.preprocessed_AdImpressions,␣usaData
cursor.execute(sql)
results = cursor.fetchall()

stateCombinations = []
count = 0

impressionList = []
clickList = []
count = 0
for each in results:
    # Get Population Array
    if (each[0] != ""):
        currentStateList = each[0].split(",")
        totalPop = 0
        for eachState in currentStateList:
            indexOfState = stateList.index(eachState)
            populationOfState = populationCount[indexOfState]
            totalPop = totalPop + populationOfState
        impressionList.insert(count, (each[1] / totalPop) * 100)
        clickList.insert(count, (each[2] / totalPop) * 100)
        count = count + 1
```

```
# For Impressions
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(impressionList) / len(impressionList)
# Plot the Histogram
counts, bin_edges = np.histogram(impressionList, bins=num_bins, weights=weight
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Population␣of␣USA␣in␣Percentage")
plt.title("CDF␣for␣Percentage␣of␣USA␣Population␣Who␣Viewed␣on␣IRA␣Published␣A
plt.grid(alpha=0.8)
#
# # For Clicks
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(clickList) / len(clickList)
# Plot the Histogram
counts, bin_edges = np.histogram(clickList, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Population␣of␣USA␣in␣Percentage")
plt.title("CDF␣for␣Percentage␣of␣USA␣Population␣Who␣Clicked␣on␣IRA␣Published␣
plt.grid(alpha=0.8)
plt.show()
```

Hierarchial Clustering and Dendogram Cut Off

```
import MySQLdb
import scipy
from scipy.cluster.hierarchy import fcluster
# DB Conn
from plotly.utils import numpy
from scipy.cluster._hierarchy import linkage
from scipy.stats._continuous_distns import frechet_l_gen
from seaborn.matrix import dendrogram
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import scipy.cluster.hierarchy as sch
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import cophenet
import matplotlib.cm as cm
from sklearn.metrics import silhouette_samples, silhouette_score

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

listOfInterests = []
```

```
count = 0

sql = "SELECT * FROM interestsTable where interestVal!='' order by interestVal
cursor.execute(sql)
results = cursor.fetchall()

for each in results:
    print(each)
    listOfInterests.insert(count, each[1])
    count = count + 1

sql = "SELECT * FROM locationOfAds where interests!=''"
cursor.execute(sql)
results = cursor.fetchall()

interestSets = []
count = 0

for each in results:
    curInterestSet = each[5]
    interestSets.insert(count, curInterestSet)
    count = count + 1

freqArr = numpy.zeros(shape=(len(listOfInterests), len(listOfInterests)))

for iLoop in range(0, len(listOfInterests)):
    currentInterest = listOfInterests[iLoop]
    curInterestIndex = listOfInterests.index(currentInterest)
    print("==================================")
    print("For␣:␣" + currentInterest)
    for jLoop in range(0, len(interestSets)):
        curInterestSet = interestSets[jLoop]
        curInterestSet = curInterestSet.split(",")
        for each in curInterestSet:
            if (each == currentInterest):
                print(curInterestSet)
                for everyElement in curInterestSet:
                    if (everyElement != "" and everyElement != currentInterest
                        currentIndex = listOfInterests.index(everyElement)
                        freqArr[curInterestIndex][currentIndex] = freqArr[curI

for i in range(0, len(freqArr)):
    tempStr = ""

    for j in range(0, len(freqArr[0])):
        tempStr = tempStr + str(int(freqArr[i][j])) + "\t"
    print(tempStr + "\n")

# generating a corelationship matrix
temp = np.corrcoef(freqArr)

for i in range(0, len(freqArr)):
    for j in range(0, len(freqArr[0])):
        # Removing redundancies
        if (scipy.math.isnan(temp[i][j])):
            temp[i][j] = -1
        if (i == j):
            temp[i][j] = -1
```

```
# print(sch.linkage(freqArr))

# G = nx.from_numpy_matrix(np.array(freqArr))
# nx.draw(G, with_labels=True)
# plt.show()
#
# print(nx.clustering(G))
# plt.scatter(nx.clustering(G))



Z = sch.linkage(temp, method='median')

dendrogram = sch.dendrogram(Z, leaf_rotation=90., leaf_font_size=8.,
                            show_contracted=True, show_leaf_counts=True,
                            get_leaves=True, no_labels=True)  # , truncate_moc
plt.ylabel("Average␣Distance")
plt.xlabel("Facebook␣Interests")
plt.title("Hierarchical␣Clustering␣of␣Facebook␣Interests")
plt.show()

print("~~~~")
temp2 = dendrogram['ivl']

for each in temp2:
    print(each)

print("~~~")
print(len(temp2))
```

```
print("~~~~~")
assignments = fcluster(Z, 5, 'distance')

print(assignments)

print("====")
noClusters = len(np.unique(assignments))
print(noClusters)
print("~~")
mainList = []
mainCount = 0
for i in range(0, noClusters):
    tempList = []
    count = 0
    for j in range(0, len(assignments)):
        if (assignments[j] == i + 1):
            tempList.insert(count, listOfInterests[j])
            count = count + 1
    mainList.insert(mainCount, tempList)
    mainCount = mainCount + 1

print("~~~~~~")
tempCount = 0
# with open('finalGroups_average_5.csv', 'w') as the_file:
#     for each in mainList:
#         tempStr = str(tempCount)
#         tempCount = tempCount + 1
#         for ever in each:
```

```
#                   tempStr = tempStr + "," + ever
#               the_file.write(tempStr + "\n")

print(len(assignments))
# Cophentic Distance
coph_dists = cophenet(Z, pdist(temp))
print(coph_dists)

# Elbow Method
last = Z[-10:, 2]
last_rev = last[::-1]
idxs = np.arange(1, len(last) + 1)
plt.plot(idxs, last_rev)

#acceleration = np.diff(last, 2)  # 2nd derivative of the distances
#acceleration_rev = acceleration[::-1]
#plt.plot(idxs[:-2] + 1, acceleration_rev)
plt.xlabel("K␣(Dendgoram␣Cut␣Off)")
plt.ylabel("Distance")
plt.show()
#k = acceleration_rev.argmax() + 2  # if idx 0 is the max of this we want 2 cl
#print("clusters:", k)

# Silouhetter Scores
listOfScores = []
listOfScores.insert(0, 0)
listOfScores.insert(1, 0)
count = 2
for n_clusters in range(2, 31):
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(Z)
    silhouette_avg = silhouette_score(Z, cluster_labels)
    listOfScores.insert(count, silhouette_avg)
    count = count + 1
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

for each in listOfScores:
    print(each)

fig, ax = plt.subplots()
plt.title("Silhouette Scores for Dendogram Cut Off")
plt.plot(listOfScores, color="blue", label="Silhouette Scores")
plt.ylabel("Silhouette Scores")
plt.xlabel('Average Distance')
ax.legend(loc='best')
plt.grid()
plt.show()
```

Focus for Clustered Interests

```
import MySQLdb
import numpy as np
from scipy.stats.stats import pearsonr
from scipy.stats.stats import pearsonr
import matplotlib.pyplot as plt
import pandas as pd
import collections
```

```
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

# List of State Abbreviations
stateAbbers = []
stateName = []
interestList = []
focusList = []
count = 0
stateLists = pd.read_csv("../states.csv", delimiter=",")
for index, row in stateLists.iterrows():
    stateAbbers.insert(count, row[1])
    stateName.insert(count, row[0])
    count = count + 1

for each in stateName:
    sql = "SELECT * FROM locationOfAds WHERE state LIKE('%" + each + "%')"
    cursor.execute(sql)
    results = cursor.fetchall()
    interestsPerAd = []
    count = 0
    for each2 in results:
        groupInterests = each2[8]
        print(groupInterests)
        groupInterests = groupInterests.split(",")
        for each3 in groupInterests:
            if (each3 != ""):
                interestsPerAd.insert(count, each3)
                count = count + 1

    print(each)
    print(interestsPerAd)
    ctr = collections.Counter(interestsPerAd)
    tempCounts = 0
    print(each)
    if (len(ctr) != 0):
        focusInterest = ""
        focusMax = ""
        for k, v in ctr.items():
            focusInterest = k
            focusMax = v
            focus = int(v) / len(interestsPerAd)
            print(focusInterest + "==>" + str(focus))
            focusList.insert(tempCounts, focus)
            tempCounts = tempCounts + 1
            print(focusInterest)
            # Update to DB
            try:
                cursor.execute(
                    """UPDATE stateList SET interGroup=%s,interGroupVals=%s W
                    (focusInterest, focus, each))
                db.commit()
            except Exception as e:
                print(e)
            break

    print("~~~~~~~~~~~~~~~~~~~")  # Plot the CDF

mean = np.average(focusList)
median = np.median(focusList)

## For Clicks
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(focusList) / len(focusList)
# Plot the Histogram
counts, bin_edges = np.histogram(focusList, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)

plt.ylabel("CDF")
plt.xlabel("Focus Values For Facebook Interests(Clustered) Corresponding To Ev
plt.title("CDF of Focus Values of All States With Respect to Facebook Interest
plt.grid(alpha=0.8)
plt.show()

print("Average: " + str(np.average(focusList)))
print("Median: " + str(np.median(focusList)))

for i in range(0, len(focusList)):
    if (focusList[i] >= 0.5 and focusList[i] < 1):
        print(focusList[i])
```

```
        print(stateName[i])

print(np.average(focusList))
print(np.median(focusList))
```

Focus for UnClustered Interests

```
import MySQLdb
import numpy as np
from scipy.stats.stats import pearsonr
from scipy.stats.stats import pearsonr
import matplotlib.pyplot as plt
import pandas as pd
import collections

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

# List of State Abbreviations
stateAbbers = []
stateName = []
interestList = []
focusList = []
count = 0
stateLists = pd.read_csv("../states.csv", delimiter=",")
for index, row in stateLists.iterrows():
    stateAbbers.insert(count, row[1])
    stateName.insert(count, row[0])
    count = count + 1

for each in stateName:
    sql = "SELECT * FROM locationOfAds WHERE state LIKE('%" + each + "%')"
    cursor.execute(sql)
    results = cursor.fetchall()
    interestsPerAd = []
    count = 0
    for each2 in results:
            interests = each2[5].split(",")
            for each3 in interests:
                    if (each3 != ""):
                            interestsPerAd.insert(count, each3)
                            count = count + 1

    print(interestsPerAd)
    print(each)
    ctr = collections.Counter(interestsPerAd)
    tempCounts = 0
    if (len(ctr) != 0):
            focusInterest = ""
            focusMax = ""
            for k, v in ctr.items():
                    focusInterest = k
                    focusMax = v
                    focus = int(v) / len(interestsPerAd)
                    print(focusInterest + "==>" + str(focus))
```

```
            break
            focusList.insert(tempCounts, focus)
            tempCounts = tempCounts + 1
            # Update to DB
            try:
                cursor.execute(
                    """UPDATE stateList SET focusInterest_PerInterest=%s,focus
                    (focusInterest, focus, each))
                db.commit()
            except Exception as e:
                print(e)
            break
    print("~~~~~~~~~~~~~~~~~~")

# Plot the CDF
## For Clicks
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(focusList) / len(focusList)
# Plot the Histogram
counts, bin_edges = np.histogram(focusList, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Focus Values of every Facebook Interest(Unclustered) Corresponding
plt.title("CDF of Focus Values of All States With Respect To Facebook Interest
plt.grid(alpha=0.8)
plt.show()
```

```
print(str(np.average(focusList)))
print(str(np.average(focusList)))
```

Topical Relevance across USA

```
# 1. Get all USA Entire by Rank
# 2. Get all #No. of ADs by Rank
# 3. Do corelation of each interest pair by rank
# 4. Plot the CDF
import MySQLdb
import numpy as np
from scipy.stats.stats import pearsonr
from scipy.stats.stats import pearsonr
import matplotlib.pyplot as plt

try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

sql = "select␣*␣from␣interestsTable␣WHERE␣interestVal!=''␣ORDER␣by␣totalTrends
cursor.execute(sql)
results = cursor.fetchall()

interestTotals = []
interestTotalsVals = []
```

```
for i in range(0, len(results)):
    interestTotals.insert(i, results[i][1])
    interestTotalsVals.insert(i, results[i][4])

sql = "select␣*␣from␣interestsTable␣WHERE␣interestVal!=''␣ORDER␣by␣totalNoAds␣
cursor.execute(sql)
results = cursor.fetchall()

zippedSort1 = list(zip(interestTotalsVals, interestTotals))

interests = []
totalAds = []
rankList = []
for i in range(0, len(results)):
    interests.insert(i, results[i][1])
    totalAds.insert(i, results[i][3])
    rankList.insert(i, 0)

zippedSort2 = zip(totalAds, interests)
corelationShipList = []
count = 0

# Assign Ranks
rank2 = 1
for each2 in zippedSort2:
    intCurrent = each2[1]
    indexSearch = interests.index(intCurrent)
    rankList[indexSearch] = rank2
    rank2 = rank2 + 1
```

```
rankList2 = []
count = 0
corleationList = []
for i in range(0, len(interests)):
    rank1 = 1
    for each2 in zippedSort1:
        if (each2[1] == interests[i]):
            rankList2.insert(count, rank1)
            count = count + 1
        else:
            rank1 = rank1 + 1

print(pearsonr(rankList, rankList2))
plt.scatter(rankList, rankList2)
plt.show()
```

Topical Relevance at State Level - Unclustered Interests

```
import os, sys

import collections
import pandas as pd
import MySQLdb
import re
from scipy.stats.stats import pearsonr
import math
import matplotlib.pyplot as plt
import numpy as np
```

```
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)

ctr = ""
sql = "SELECT␣*␣FROM␣stateList"
cursor.execute(sql)
results0 = cursor.fetchall()
stateList = []
corrVal = []
mainCount = 0

pvalue = []

# Fetch All data
for each0 in results0:
    # Get all the Ads & corresponding #Ads in current State
    sql = "SELECT␣*␣FROM␣locationOfAds␣WHERE␣state␣LIKE('%" + each0[1] + "%')"
    cursor.execute(sql)
    results1 = cursor.fetchall()
    listOfInterests_FromState = []
    count = 0
    for each1 in results1:
        allInterests = each1[5]
        splitInterests = allInterests.split(",")
        for eachInterest in splitInterests:
```

```
        if (eachInterest != ""):
            listOfInterests_FromState.insert(count, eachInterest)
            count = count + 1

# Ranking the list of interests
ctr = collections.Counter(listOfInterests_FromState)
# Get Google Trends Data
googleTrendsInterests = []
googleTrendsAvg = []
count = 0
sql = "SELECT␣*␣FROM␣stateTrends␣WHERE␣state='" + each0[1] + "'␣ORDER␣BY␣a
cursor.execute(sql)
results2 = cursor.fetchall()
for each2 in results2:
    googleTrendsInterests.insert(count, each2[2])
    googleTrendsAvg.insert(count, each2[3])
    count = count + 1

# RANK ; CORELATE
rank1List = []
rank2List = []
rank1 = 1
rank2 = ""
count = 0
for k, v in ctr.items():
    curInterest = k.lower()
    try:
        indexOfGoogleTrends = googleTrendsInterests.index(curInterest)
        rank2 = indexOfGoogleTrends + 1
        except Exception as e:
            indexOfGoogleTrends = -1
            rank2 = -1
        rank1List.insert(count, (count + 1))
        rank2List.insert(count, rank2)
        count = count + 1

    pearSonVal = pearsonr(rank1List, rank2List)
    if (not math.isnan(pearSonVal[0])):
        stateList.insert(mainCount, each0[1])
        corrVal.insert(mainCount, pearSonVal[0])
        pvalue.insert(mainCount, pearSonVal[1])
        mainCount = mainCount + 1

for i in range(0, len(stateList)):
    print(stateList[i] + "==>" + str(corrVal[i]))
# CDF
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(corrVal) / len(corrVal)
# Plot the Histogram
counts, bin_edges = np.histogram(corrVal, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
plt.xlabel("Pearson␣Correlationship␣Values␣of␣Interests(Unclustered)␣and␣Googl
plt.title("CDF␣of␣Correlationship␣Between␣Google␣Trends␣and␣Interests(Uncluste
plt.grid(alpha=0.8)
plt.show()


print("avg␣p␣value␣:␣"+str(np.average(pvalue)))
print("median␣p␣value␣:␣"+str(np.median(pvalue)))
```

Topical Relevance at State Level - Focussed Topical Groups

```
import os, sys


import collections
import pandas as pd
import MySQLdb
import re
from scipy.stats.stats import pearsonr
import math
import matplotlib.pyplot as plt
import numpy as np
import csv
pvalue = []
try:
    db = MySQLdb.connect("127.0.0.1", "root", "", "tempUsa")
    # Initialise Cursor
    cursor = db.cursor()
except Exception as e:
    print(e)
# Getting the interest groups
f = open("finalGroups_average_5.csv", 'r')
listOfInterests = []
count = 0
with f:
    reader = csv.reader(f)
    for row in reader:
        tempList = []
        countTemp = 0
        for everElement in row:
            tempList.insert(countTemp, everElement)
            countTemp = countTemp + 1
        listOfInterests.insert(count, tempList)
        count = count + 1

ctr = ""
sql = "SELECT * FROM stateList"
cursor.execute(sql)
results0 = cursor.fetchall()
stateList = []
corrVal = []
mainCount = 0



def remData_Ensure():
    # Ensure ALL google trends data is there
    for each_n in listOfInterests_FromState:
        sql = "SELECT * FROM stateTrends WHERE state='" + each0[1] + "' AND in
        cursor.execute(sql)
        results_n = cursor.fetchall()
        if (len(results_n) == 0):
            print(each0[1] + "==>" + each_n)
```

```
# Fetch All data
for each0 in results0:

    interestGroup = each0[9]
    interestByGroup = listOfInterests[interestGroup]

    # Get all the Ads & corresponding #Ads in current State
    sql = "SELECT * FROM locationOfAds WHERE state LIKE('%" + each0[1] + "%')"
    cursor.execute(sql)
    results1 = cursor.fetchall()
    listOfInterests_FromState = []
    count = 0

    for each1 in results1:
        allInterests = each1[5]
        splitInterests = allInterests.split(",")
        for eachInterest in splitInterests:
            if (eachInterest != ""):
                for each_n in interestByGroup:
                    if (each_n == eachInterest):
                        listOfInterests_FromState.insert(count, eachInterest)
                        count = count + 1

    # Ranking the list of interests
    ctr = collections.Counter(listOfInterests_FromState)
    # Get Google Trends Data
    googleTrendsInterests = []
    googleTrendsAvg = []
    count = 0
```

```
sql = "SELECT * FROM stateTrends WHERE state='" + each0[1] + "' ORDER BY a
cursor.execute(sql)
results2 = cursor.fetchall()

for each2 in results2:
    for each_n in interestByGroup:
        if (each_n == each2[2]):
            googleTrendsInterests.insert(count, each2[2].lower())
            googleTrendsAvg.insert(count, each2[3])
            count = count + 1

# rank and corelate
rank1List = []
rank2List = []
count = 0
for k, v in ctr.items():
    curInterest = k.lower()
    rank1List.insert(count, (count + 1))
    flag = 0

    for zj in range(0, len(googleTrendsInterests)):
        if (googleTrendsInterests[zj].lower() == curInterest):
            flag = zj + 1
            rank2List.insert(count, flag)

    count = count + 1
    # Flag Check
    if (flag == 0):
            print(each0[1])
            print(curInterest)
            print(googleTrendsInterests)
            print("~~~~~~~~~~~~~~~~~~~~~~~~~~")

    # rank and corelate
    print(rank1List)
    print(rank2List)
    print("==================")
    pearSonVal = pearsonr(rank1List, rank2List)
    if (not math.isnan(pearSonVal[0])):
        stateList.insert(mainCount, each0[1])
        corrVal.insert(mainCount, pearSonVal[0])
        pvalue.insert(mainCount, pearSonVal[1])
        mainCount = mainCount + 1

for i in range(0, len(stateList)):
    print(stateList[i] + "==>" + str(corrVal[i]))

# CDF
num_bins = 10
fig, ax = plt.subplots()
# Normalising the PDF
weights = np.ones_like(corrVal) / len(corrVal)
# Plot the Histogram
counts, bin_edges = np.histogram(corrVal, bins=num_bins, weights=weights)
cdf = np.cumsum(counts)
plt.plot(bin_edges[1:], cdf)
plt.ylabel("CDF")
```

```
plt.xlabel("Pearson␣Correlationship␣Values␣of␣Focussed␣Topics␣and␣Google␣Trend
plt.title("CDF␣of␣Correlationship␣Between␣Google␣Trends␣and␣Focussed␣Topic␣of␣
plt.grid(alpha=0.8)
plt.show()
print("avg␣p␣value␣:␣"+str(np.average(pvalue)))
print("median␣p␣value␣:␣"+str(np.median(pvalue)))
```

Ad Objective	Objective	Platforms	Ad Formats
Awareness	Brand Awareness	Facebook Instagram Messenger	Single Image Single Video Carousel Slideshow
	Reach	Facebook Instagram Messenger	Single Image Single Video Carousel Slideshow
Consideration	Traffic	Facebook Instagram Messenger Audience Network	Single Image Single Video Carousel Slideshow Collection
	App Installs	Facebook Instagram Messenger Audience Network	Single Image Single Video Carousel Slideshow
	Engagements	Facebook Instagram	Single Image Single Video Slideshow
	Video Views	Facebook Instagram Audience Network	Single Video Carousel Slideshow
	Lead Generation	Facebook Instagram Messenger	Single Image Single Video Carousel Slideshow
	Messages	Facebook Instagram Messenger	Single Image Single Video Carousel Slideshow
Conversions	Catalogue Sales	Facebook Instagram Messenger Audience Network	Single Image Carousel
	Conversions	Facebook Instagram Messenger Audience Network	Single Image Single Video Carousel Slideshow Collection
	Store Visits	Facebook	Single Image Single Video Carousel Slideshow Collection

Table 9: Description of all available objectives of Facebook Ad Campaigns