Winter 2023

Combating Fake News: A Gravity Well Simulation to Model Echo Chamber Formation In Social Media

Jeremy E. Thompson
Dartmouth College, jeremy.e.thompson.th@dartmouth.edu

COMBATING FAKE NEWS: A GRAVITY WELL SIMULATION TO MODEL ECHO CHAMBER FORMATION IN SOCIAL MEDIA

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

in

Engineering Sciences

by Jeremy E. Thompson

Thayer School of Engineering
Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

OCTOBER 2023

Examining Committee:

Chairman_____
Dr. Eugene Santos, Jr.

Member_____
Dr. George Cybenko

Member_____
Dr. Vikrant S. Vaze

Member_____
Dr. John Korah

_____

F. Jon Kull, Ph.D.

Dean of the Guarini School of Graduate and Advanced Studies

Abstract

Fake news has become a serious concern as distributing misinformation has become easier and more impactful. A solution is critically required, but choosing the right solution is perhaps just as critical. One solution would be to ban fake news, but that approach could create more problems than it solves, and would also be problematic from the beginning, as it must first be identified to be banned. We propose a method to automatically recognize suspected fake news, and to provide news consumers, as well as researchers, historians, and journalists, with more information as to its veracity. It is suggested that fake news is comprised of two primary components: premises and misleading content. A fake news piece can be condensed down to a collection of premises, which may or may not be true, and to various forms of misleading material, including biased arguments and language, misdirection, and manipulation. Misleading content can be exposed for whatever biases it contains, regardless of the intent of the author. While this framework can be valuable, its utility may be limited by the rapid improvement in artificial intelligence, which can be used to alter fake news strategies at a rate that could exceed the ability to update the framework. Therefore, more immediately we propose a model for identifying echo chambers, which are widely reported to be havens for fake news producers and consumers. We simulate a social media interest group as a gravity well, through which we can identify the online groups most postured to become echo chambers, and thus a source for fake news consumption and replication. This echo chamber model is supported by three pillars related to the social media group: technology employed, topic explored, and confirmation bias of group members. The model is validated by modeling and analyzing 19 subreddits on the Reddit social media platform. Contributions include a working definition for fake news, a

framework for recognizing fake news, a generic model for social media echo chambers including three pillars central to echo chamber formation, and a gravity well simulation for social media groups, implemented for 19 subreddits.

Acknowledgements

First and foremost, I owe an extreme debt of gratitude to my advisor, Dr. Eugene Santos, Jr. I would not have even ventured forth on this journey without his prompting and sponsorship. His guidance and perception helped me find my way over and through several obstacles and challenges. Moreover, he managed to maintain confidence in my ability to soldier on and succeed, even and especially when I doubted myself. I genuinely have difficulty conceiving of any other advisor getting me to the finish line.

Additionally, I am genuinely grateful to my committee, Dr. George Cybenko, Dr. Dr. Vikrant S. Vaze, and Dr. John Korah, who came through with insightful advice and improvements both before and after my defense. Their input undoubtedly improved the quality of this dissertation and granted me the gift of pride in my final product.

Further, I would certainly be remiss if I failed to acknowledge the financial support provided by the Air Force Office of Scientific Research and their grants which helped fund my research. Without such support, the advancement of basic research and science would be far more difficult to accomplish.

The list of colleagues who kept me going over the years is quite long. I will attempt to give credit where credit is due, but I am nearly certain I will fall short. First among these would have to be Jacob Russell, who amazingly managed to overlap my extended duration at Dartmouth significantly. He frequently provided encouragement when needed, and alternative perspectives and input when desired. I cannot begin to recall the number of instances where Jacob advised on programming, configuration, proofing, and version control. Over the years, many other colleagues have helped me along the road to success:

Fei, Yuki, Chase, JT, Jake, Suresh, and Mani. Each has had a unique and positive effect on me and my story, for which I am grateful. Many others are owed thanks as well, but I trust they know their contributions are appreciated if not highlighted.

Thanks also to the undergraduate interns that contributed time and effort towards projects reviewed in the Explorations section: April Liu, Allyssa Austin, Divya Kopalle, and David Nesbitt. Additionally, thank you to the Women in Science Program, the Claire Booth Luce Fellowship, and the First Year Research in Engineering Experience program.

Finally, I would be remiss if I did not mention family. My sisters Sheryl and Jennifer have provided encouragement and advice over the years. Likewise, my (grown) children Chris and Quin have been unfailing in cheering me on through the many years of my PhD pursuit, never doubting I would get there in the end. Unexpectedly, I also have a new wife to thank for providing support, distractions, sustenance, and affection these last few years when my doubts about finishing were peaking. Ling provided me with the energy and determination I needed to push through to success.

Table of Contents

List of Tables

List of Figures

List of Acronyms

analysis of variance	ANOVA
application programmer interface	API
Bayesian Knowledge Bases	BKBs
betweenness centrality	BC
Bidirectional Encoder Representations from Transformers	BERT
bidirectional long short-term memory	bi-LSTM
closeness centrality	CC
convolutional neural networks	CNNs
document graphs	DGs
generative pre-trained transformer	GPT
instantiation node	I-node
Islamic Court Union	ICU
Islamic State in Iraq and Syria	ISIS
large language model	LLM
Linguistic Inquiry and Word Count	LIWC
long short term-memory	LSTM
maximum entropy	MaxEnt
mean average percent error	MAPE
normalized mean absolute error	NMAE
random variable	RV
Perspective Comment Analyzer	PCA
support node	S-node
support vector classification	SVC
support vector machine	SVM

technology modifier	TM
topic source modifier	TSM
term frequency/inverse document frequency	TFIDF

1. Introduction

It is no secret that fake news has become a serious concern [1]–[3]. Some note that it has become a danger to democratic society [4]–[9]. For a democracy to function healthily, it must rely on an educated, well-informed voting public. Differences in opinions, values, and priorities will naturally exist, but those differences can only be legitimately debated and resolved if the voting public has access to information, and if they have some reasonable means for distinguishing the validity and accuracy of that information. This is not a new problem, of course. Voters in the U.S. have been swayed and manipulated by the newspaper industry essentially from the beginning [10, Ch. I], and fake news in the form of propaganda has been traced all the way back to the Roman Empire [11]. However, with the advent of social media, access to a vulnerable public has become widespread, and distributing misinformation has become a trivial endeavor with noticeable effect [1], [12].

The threat is not just to the democratic process, but to a functioning, safe society in general. Instances of panic and violence have been prompted by the spread of fake news, such as the shooting at Comet Ping Pong spawned from a story that the pizza restaurant in Washington, D.C. was the haven of a child sex ring [13], a hoax surrounding a fictional attack on a chemical plant in Louisiana by Islamic State in Iraq and Syria (ISIS) [14], lynch mobs in India as the result of false videos spread on WhatsApp [15], and the massacre of between 86 and 238 individuals in Nigeria fueled by fake news generated on Facebook [16]. More recently, threats to public health have emerged in the guise of rampant misinformation surrounding the COVID-19 pandemic [17]–[19].

Social media has been under pressure to remedy the spread of fake news, with only modest results. Mark Zuckerberg, CEO of Facebook, was called to testify by the U.S.

House of Representatives. In his testimony [20], Mr. Zuckerberg acknowledged the problem of fake news, specifically with respect to U.S. election interference, and proposed essentially two approaches to remedy the situation. First, he asserted that Facebook would increase efforts to identify and remove fake accounts, which are suspected of being used by actors, including foreign nations, to spread content aimed at affecting election outcomes. Second, he declared that advertisers wishing to run political or issue ads must be registered, with their identity and location confirmed. This effort at advertising transparency online is not currently required by U.S. law and remains a glaring loophole in current political advertising legislation. While admittedly this is a difficult problem to address, the efficacy of Facebook's approach remains to be determined. Some recent indications are encouraging [21]. However, one study [22] found "Facebook's current enforcement appears imprecise: 61% more ads are missed than are detected worldwide, and 55% of U.S. detected ads are in fact non-political."

Similarly, when YouTube was recently challenged concerning its platform being a haven for the spread of conspiracy theories, Susan Wojcicki, then CEO of YouTube, declared her company's approach would be to accompany suspect videos with a link to more reliable information, such as Wikipedia [23]. Beyond the concerns about Wikipedia being able to sustain its current, sometimes questioned, level of reliability as mentioned in the cited article, it seems conspiracy theories continue to flourish on YouTube, [24], [25].

It should be evident that a remedy is critically required, but choosing the right remedy or remedies is perhaps just as critical. An impulsive answer might be to ban fake news. It is not difficult to imagine how banning fake news could backfire and produce more problems than solutions. Censorship in any form must be approached with temerity and

caution. The remedy advocated here is to provide more information to the public, not less. Ultimately, an "education for freedom" as proposed in Brave New World Revisited [26] could be the panacea for a healthy, democratic society. As Huxley notes, education for freedom should be "an education first of all in facts and in values…" However, to aim more realistically in the short-term, a method for recognizing suspected fake news, and providing news consumers with the tools to consume a healthier diet of information, should they choose to, would be a significant step in the right direction.

In order to accomplish this, we propose that a framework be applied to recognize misleading information in news stories. The most essential insight into the approach proposed here is that fake news can be broken down into critical components which point to its misleading character. The macro components are premises and misleading content.

While the premises are initially planned to be simply identified and highlighted, it is possible that methods could be employed to investigate the validity of the premises. That approach is deferred to other research, as it has a danger of spiraling into a subjective dispute based on political, moral, or personal values. Instead, it is suggested that highlighting the premises will serve well to allow a reader or researcher to gauge how firmly any arguments begin. The associated misleading content could then be recognized by contextual indicators, including misdirection, bias, and manipulation. Based on current literature searches, this is not only a promising approach to the problem, but also a unique one.

The uniqueness stems from the insight of discovering what makes up fake news, not just attempting to find an automated rule that will separate out fake from real. With no

understanding of the phenomena of fake news, nor of the rule, any automated solution would be a temporary one, as fake news producers adapt and change their methodology. Instead, the fake news framework as outlined here applies the concept of methodological reductionism [27], striving both to understand what comprises fake news and to apply that understanding to recognizing fake news. The framework is not a static solution. As our understanding of fake news grows, that understanding can be added to the framework to increase its efficacy. The two primary components of the framework—premises and misleading content—are unlikely to change. However, new indicators for misleading content may be identified, which can then be readily integrated into the framework.

However, to aim more realistically in the short-term, a method for recognizing suspected fake news, and providing news consumers with the tools to consume a healthier diet of information, would be significant steps in the right direction. We will relate our investigations into forming this framework here as it informs the problem and retains its merit, but ultimately decided that a more effective way forward, at least in the short-term, if not also for the long-term, due to its resiliency to intentional obfuscation and manipulation by malefactors, involves modeling consumers of fake news and where they congregate rather than modeling or recognizing fake news itself.

As stated, after some investigations into the construction of the fake news framework, the immediate remedy we propose as an effective and dynamic solution to combatting fake news is to determine a method of recognizing the breeding grounds for fake news. While keeping up with the latest techniques employed by the originators of fake news is problematic, especially for a small team, we suggest the real concern lies with the adoption and repetition of the fake news by social media users. Researchers report much of this

process occurs in social media echo chambers [28], [29]. By echo chamber in this context, we mean the oft cited "bounded, enclosed media space that has the potential to both *magnify* the messages delivered within it and insulate them from rebuttal" (emphasis added) [28, p. 76].

Thus, to get to the crux of the problem, we propose identifying the environs where fake news can flourish as a key step to informing the public as to their exposure to potentially unhealthy, unbalanced, and unexamined discourse. To effectuate that goal, we endeavor to develop a model that can identify the hallmarks of an echo chamber by the behavior and posts of the members of a social media group, without the need to determine the precise veracity of every post, thus providing a more universal solution for multiple social media platforms, as well as for multiple subject domains. Ultimately, this approach could be combined with other analyses to inform readers of their exposure to fake news.

Our primary hypothesis is that an interest group on a social media platform can be effectively modeled as a gravity well [30], a similitude for the gravitational pull exerted by a large mass in space. A gravity well is effectively a metaphor for Einstein's geodetic effect, commonly visualized as a bowling ball distorting the planar surface of a trampoline due to its weight [31]. An illustration of this concept can be found in Figure 1-1[1]. The analogy here is that an echo chamber in a social media interest group essentially *captures* its audience, with elements of the social media platform—the precise topic of the forum

[1]This Photo by Unknown Author is licensed under CC BY-SA-NC

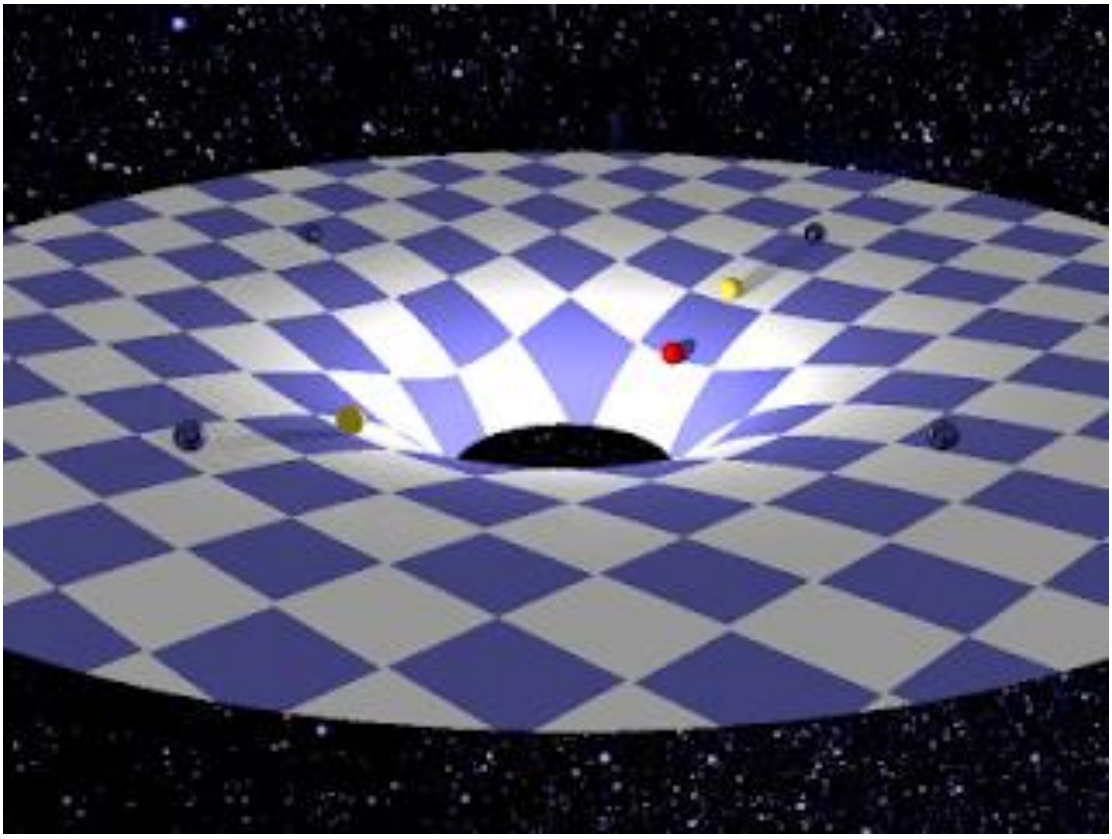

Figure 1-1: Example gravity well

and characteristics of the audience constituting the attractive force that holds users within the grip of the gravity well. As indicated by the title of our work, we further theorize that an echo chamber could then be detected by particular aspects of that gravity well.

Contributions of this research are our working definition for fake news and our proposed framework for recognizing fake news. Further contributions include the

definition of a generic method for modeling echo chambers, to include the proposal of three pillars essential for an environment conducive to the formation of echo chambers. Moreover, this proposed model employs a novel approach using the concept of a gravity well to represent social media groups in a multiagent simulation. This generic model aims

to be equally applicable to multiple social medial platforms. More immediately, we demonstrated the use of our echo chamber simulation to nineteen different subreddits within the Reddit social media platform.

In the remainder of this document, the research is discussed in greater detail. Background information is provided, followed by a section describing Steps Toward Fake News and Echo Chambers. We next outline the Fake News Model Formation and Exploration. With that foundation in place, we introduce the concepts of Echo Chambers and Gravity Wells, followed by Gravity Well Simulation, Results, and Analysis. Lastly, we address Conclusions and possibilities for Future Work.

2. Background

In this section, some topics central to addressing the problem of fake news will be reviewed. First, the profession of journalism will be discussed, including the aspects of journalism that have, to some extent, kept fake news historically in check, which unregulated or unprofessional writings arguably do not. Next, fake news will be analyzed, including common definitions adopted thus far and a treatment on why a novel definition is required. Finally, a discussion on the automatic detection of fake news will naturally segue to a discussion on the concept of echo chambers, where relevant research is reviewed.

2.1. Journalism

Unofficial news sources are springing up worldwide, causing journalists and traditional news organizations to struggle to remain relevant [32]–[34]. With so much of fake news being false news intentionally crafted to resemble real news [3], [12], [35]–[39], it is beneficial to review some essential aspects of professional journalism.

2.1.1. Gatekeeping

Gatekeeping is conceptually one role that a newspaper editor has traditionally filled. The duties of an editor are to select (or reject) news stories based on the priorities of the publication. This is not to say that an editor is the only gatekeeper in the news media [40]. There are many gates through which a story must pass before it appears in print (or in digital). However, editors are one of the most powerful and visible gatekeepers. This is vividly illustrated in [41], where Dr. White analyzes the stories accepted and rejected by a newspaper editor, and the reasons given for each. The reasons for rejecting an article vary from the quality of the writing to an overabundance of stories on the topic, to the story being dull or trivial. There are arguments for and against this type of formal gatekeeping,

with advocates arguing that without this sort of gatekeeping, we would see something like what we are seeing on social news sites—an overabundance of fake news. On the contrary, opponents argue that gatekeepers determine what is newsworthy, and often what is newsworthy is determined by those in power [42, p. 101] according to the 'official line'. This has been most evident during times of war, when propaganda comes into primacy [43, Sec. One].

With the advent of non-traditional news reporting, most gatekeeping has disappeared, and everyone must be their own gatekeeper. This can be advantageous, as much news that was historically unavailable has now become a trove of information for any seeking it. However, as the advocates of formal gatekeeping expected, false and dishonest news reporting is also flourishing, and the public has not been well-equipped to sort through volumes of misinformation for the occasional nugget of truth. Expectations have been set for social media to address this problem [23], [38], [44], but others argue that asking private corporations to take on the mantle of gatekeepers is fraught with problems, including the self-interest of corporations, the likely inefficacy of that approach in the end, and threats to freedom of expression [45].

2.1.2. Objectivity

Objectivity as it applies to journalism is tied to the idea that, in order to report the news as it is, without being colored with bias or prejudice, one must be impartial, balanced, factual, fair, and detached [46]. It is aimed at allowing reported news to reflect a common world view, upon which all can agree [47, Ch. 1]. The objectivity norm in journalism was developed early in the 20th century in the U.S. and became a central tenet of journalism in the Western world [46]. While it commanded respect for much of the 20th century, it began

to suffer much criticism later in the century, being described as an unattainable and even dangerous ideal [48]. This criticism began to invade the world of journalism, to the point where the Society of Professional Journalists removed the term from its code of ethics in the 1990s [49, p. 5].

Despite objectivity's rise and fall [42, Ch. 3], it must necessarily play some part in the handling of fake news. Though arguments have been presented that a new generation of news consumers are not rejecting truthful news [33], but only desire a more opinionated form absent objectivity, this appears to be a more unobtainable truth than the dream of objectivity. As the authors of [50] show, not only is a segment of the users of social news sites drawn to hoax news, but it is also possible to predict hoax news based on which users "liked" them on Facebook. The good news here is that not all news consumers are rejecting objectivity and gravitating to fake news. A possible approach to remedy this situation is to encourage a healthier diet of news consumption, just as to alleviate the growing health problems in the US, promoting a healthier diet of food consumption could work wonders.

2.2. Fake News

To fully explore the motivation for this research, we must first discuss the central concern, that of fake news. Firstly, the motivation for studying this problem will be discussed. As indicated previously in the Introduction, the prevalence of fake news and its influence has become a prominent concern across the globe [1], [2], [5], [12]–[16], [51]. Moreover, in the United States, the seriousness of the issue cannot be better illustrated than the findings of multiple U.S. government agencies assessing that a foreign government (Russia) applied cyber warfare tools and techniques to undermine the democratic process [35]. Much of that effort was focused on the spread of false information on social media platforms. While the

U.S. agencies made no conclusions as to the efficacy of those efforts, other researchers have determined that fake news was voraciously read by some sectors of the population [2], some noting a definite impact on the elections from same [1], [52], and one finding fake news was more widely shared on Facebook during the election than real news pieces [53]. Perhaps even more disturbing, a recent study by Stanford researchers [54] found that 7,804 young people in schools across twelve states demonstrated a *bleak* ability to reason about information they encountered on the internet.

It is important to make clear what is meant by fake news. In some literature, fake news begins with the political satire famously demonstrated by the television program *The Daily Show* [37]. While that program often labeled itself fake news, the news presented there was, on the whole, quite accurate. The show provided entertainment by ridiculing the news and the players involved. While some would certainly argue the show exhibits a (liberal) bias, the fact that it does not try to pass itself off as a legitimate news source suggests that it is not of primary concern to the current research.

Similarly, the authors in [37] lump news parodies in their typology of fake news. A prime example of this would be the parody site *The Onion*[2]. Again, this type of source, one primarily for entertainment and openly declared as such, is not of interest to this research. Of course, there are some who might be taken in by humor news, but the cure for that resides somewhere other than technological approaches to rooting out misleading news

[2] https://www.theonion.com/

content. This definition of fake news, excluding satire and parody, is more in line with the explanation provided in [52]. Thus, we arrive at our working definition of fake news: "False or misleading content represented as news, regardless of the intent to deceive."

The emphasis we focus on here is on the claim of "news". If an article presents itself as news, yet has an agenda to persuade rather than inform, it would be considered fake news. An article could be completely true and factual, but if it, through omission, makes no attempt at a balanced discussion or debate, it would fall within the fake news spectrum defined for the current research. For this reason, only articles of some length will be considered. As shorter headlines and breaking news may only have the briefest discussions on a topic, a balanced discussion would be difficult. This is not to say that such short articles, tweets, and headlines could not be fake, but that they do not fall within the scope of analysis for this work. It is conceivable, though, that a collection of tweets from a single author could be analyzed using the framework. Such an effort is reserved for later consideration. Many examples of misleading and completely false sensationalist headlines exist, but there have also been efforts to address this problem [55], [56]. Simple fact-checking could be an effective bastion against such limited pieces, especially with the assistance of network analysis for automated fact checking [57], as well as source reliability measures [58], both of which have become common, though laborious, approaches to identifying fake news.

2.3. Automatic Fake News Detection

Automatic fake news detection is a relatively new field of study, coming to the forefront in concert with the recent concerns over the negative effects of unreliable news spreading via social networks [1], [12]–[16], [59]. Many researchers have been attempting to find

solutions to this problem. A brief overview of some of the most successful approaches is provided below. Of interest for the current research are studies which attempt to classify text based purely on the content of the text, as other metadata is often not available or is exclusive to a particular social network.

In [60], the author introduces a publicly available fake news dataset called LIAR[3] that could be advantageous. The database is comprised of 12,836 short statements from the PolitiFact website[4]. These statements have been classified on a discrete six-point spectrum from "pants on fire" to "true", with the labels "false", "barely true", "half true", and "mostly true" falling between. The author then tested several classification approaches, reporting the greatest success with convolutional neural networks (CNNs), with a reported accuracy of 0.270 for the test set, compared to an accuracy of 0.208 by exclusively selecting the majority label. The other models, including support vector machine (SVM), logistic regression, and bidirectional long short-term memory (bi-LSTM) networks models, fell between these two extremes. The author also found some modest improvement by including select metadata (subject, speaker, job, state, party, context, and history) related to the original texts, but the improvement was modest indeed, with an accuracy 0.274.

Subsequent studies have methodically improved on this baseline with the advent of more capable classification models. Of recent note, through the use of Google's Bidirectional Encoder Representations from Transformers (BERT), the authors of [61]

[3] http://www.cs.ucsb.edu/~william/data/liar_dataset.zip

[4] http://www.politifact.com/

reported a classification accuracy of 0.70 on the LIAR dataset. This reflects the giant strides made recently in large language models (LLMs), which we will address later.

The authors of [62] conducted some interesting analysis into the uses of particular categories of words by news articles classified as either "trusted", or one of three types of fake news: "satire"; "hoax"; and "propaganda". According to the fake news definition identified in the current study, the primary interest is in hoax and propaganda news, but the ability to recognize or at least separate out satire is also of value. The authors made use of the Linguistic Inquiry and Word Count (LIWC) lexicon, as well as sentiment, hedging, and intensifying lexicons, all of which could prove useful in the fake news framework described in Section 4. The authors also applied what they learned to predicting the truthfulness of news articles. Of particular interest is their application of classifiers to a PolitiFact[5] database which labels news in a discrete spectrum from "pants on fire" to "true", as in [60]. They report an F1 score of 0.20 for the long short-term memory (LSTM) classifier, compared with a score of 0.06 for the majority baseline. Oddly, when they added LIWC feature vectors as input, the results suffered for the LSTM solution, scoring only 0.19, despite the other models, naïve Bayes and maximum entropy (MaxEnt), improving markedly. The best result was that of the MaxEnt model with LIWC, with an F1 score of 0.22. Unfortunately, direct comparison with [60] is not possible since that author reported classification accuracy and not F1 scores.

[5] http://www.politifact.com/

Another investigation of interest is the one detailed in [63], as the authors also attempt to find language identifiers for misleading content. The authors focused on Twitter as a data source and compiled a corpus of more than 130 thousand retweets[6], which they then labeled according to the original tweeter's account veracity—either verified (by Twitter[7]) or suspicious (with the help of two publicly available tools that annotate suspicious Twitter accounts[8]). The authors focused on two linguistically infused neural network models, LSTM and CNN, as compared to two logistic regression models employing term frequency/inverse document frequency (TFIDF) features or Doc2Vec vectors [64], [65]. It is perhaps no surprise that the LSTM and CNN methods both outstripped the logistic regression models on a binary classification task, both returning an accuracy of 0.95 and precision of 0.99 when utilizing not just the tweet text and linguistic cues, but also Twitter user network information. While this looks impressive on the surface, what is not clear is how much overlap exists between the text of the original tweets and the retweets. Additionally, the subject matter of the tweets was specifically scoped by including tweets only from one week before and after the Brussels bombing occurring on 22 March 2016. The primary interest for the current research is the set of linguistic cues the authors employed in their study, including bias, subjectivity, psycholinguistic, and moral foundation cues. The authors performed a detailed linguistic analysis of the linguistic

---

[6] http://www.cs.jhu.edu/~svitlana/data/fakenews_dataset.zip

[7] Twitter verification has changed substantially since this research was published: https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts

[8] http://www.propornot.com/p/the-list.html and http://www.fakenewswatch.com/

markers and reported their findings in detail. Their aim was to help explain the results of their neural network models, which of course do not provide explanations for outcomes given. The authors' linguistic analysis could prove beneficial to our fake news framework.

2.4. Echo Chambers

While the originators of fake news may be at fault for the production of misinformation, perhaps what is most problematic about fake news is that it appears to thrive and readily replicate in today's society. One possible cause for this is the proliferation of phenomena known as echo chambers on modern social media platforms. As previously referenced, we can define an echo chamber as a "bounded, enclosed media space that has the potential to both *magnify* the messages delivered within it and insulate them from rebuttal" (emphasis added) [28, p. 76]. That definition perhaps intimates why social media would be a breeding ground for such situations, as social media users often elect to insulate themselves.

One of the earliest references to social media and echo chambers identifies that very propensity concerning blogs [66]. The authors note the prevalence of comment agreement relative to disagreement with the original blogger, finding that ratio varied from 2 to 1 upwards to 9 to 1. This study focused on the question of whether blogs might be considered echo chambers, and whether any such echo chambers were more likely to form according to the genre of the blog. The most disturbing result in the context of this thesis is that among the genres with the highest ratios of agreement to disagreement—i.e., the strongest echo chambers—was the group of political blogs, with a ratio of 9 to 1.

Many other studies of echo chambers and social media have followed [29], [67]–[80], making it clear that this is a concerning trend, and one the study of which that society

might benefit. To learn more about echo chambers in social media, we must first be able to recognize them. Many researchers have already attempted this task, which is what we shall review in the remainder of this section.

2.4.1. Studies of interest

Perhaps the most straightforward approach would be to attempt to identify echo chambers using the actual content within the potential echo chamber. This is the approach explored by the authors in [75], combining content-based sentiment analysis to identify argument stance and the emotions elicited by a subject. Their combined approach was successful in scoring highest the leftmost- and rightmost-leaning news fan pages. While this approach shows some promise, it was only trialed on ten fan pages, so more evidence is needed to determine its efficacy.

In another approach, the authors of [80] suggest that applying a community detection strategy to tweets can be effective in identifying echo chambers. Both the topology of relationship networks and conversation graphs are employed to identify pockets of polarization within discussions related to COVID-19. Ultimately, the investigators concluded that, while the addition of semantic information may have improved identification of echo chambers, further research is required to determine the vibrancy of the echo chambers identified. This unfortunately is a harbinger of a similar limitation we have found with our own research.

Yet another approach was tried in [77]. The authors strove for a general framework for identifying echo chambers across multiple online social networks. They focused on topologies but from a macro-level. They incorporated other indicators of echo chambers

which may be found across a variety of social medial platforms, specifically: controversial topics, user posts, and user comments. The authors identified the existence of several echo chambers on the Reddit platform. However, their results are mixed and do not correspond clearly with another prominent echo chamber research effort [73]. Their mixed results only indicate that the problem is far from solved. Furthermore, as noted by the authors and relevant to the research reported here, there is no standard for determining the veracity of echo chamber detection. More research is required on all fronts to meet this challenge.

In direct contrast to the previous effort, the authors of [73] concluded the Reddit platform was essentially absent of echo chambers, at least with respect to two political camps. Their methodology involved reconstructing the political interaction networks of users within Trump and Clinton subreddits to determine whether there were substantial interactions between these opposite political groups. The authors found significant asymmetric heterophily and a preference for cross-cutting political interactions between them. This is counter to expectations set by the echo chamber narrative, but once again, ground truth is difficult to establish. Moreover, this was a study limited to only two potential echo chambers.

2.4.2. Simulations

In addition to finding ways to aggregate and analyze real data from potential echo chambers, sometimes it can be beneficial to produce a simulation that provides insights into the data used to seed the simulation. One such effort can be found in [76], where Törnberg models social media networks and searches for associations between echo chambers and viral misinformation spread. To that end, the author characterizes viral misinformation spread as the diffusion of complex contagions. By examining data from the

perspective of echo chambers being defined by polarization of both opinions and networks, the author was able to identify potential echo chambers, then relate those chambers (network clusters) to the virality of a contagion (misinformation spread). In the end, the study concludes that the identified echo chambers may be linked to misinformation spread. While the research adds to existing literature on network diffusion, the entire simulation is based on synthetic network data generated to study the proposed characteristics echo chambers may have in social networks. At this juncture, there is no tie to real-world observations. Application to real social network data remains to be demonstrated, assuming availability of the data required to generate this simulation based on real-world social media networks.

In another research effort based on simulation, the authors of [81] propose a model that focuses on the dynamics of radicalization as a reinforcing mechanism leading to extreme opinions from otherwise moderate initial conditions. Their model focuses on the influences of homophily and heterogeneous activities on echo chamber formation. The authors aim to reduce the contradictions exposed between observations of real-world social media networks and the predictions of classical models of opinion dynamics. Once again, this is purely a theoretical simulation based on synthesized data. However, the research has significant findings regarding the similarity of their theoretical results to actual data from polarized political discussions on Twitter. The similarities between the model data and the political discussions are striking. Further research could prove helpful in identifying the characteristics of echo chambers in online social media platforms.

Speaking of further research, the authors of [82] employed the simulation in [81] to explore a bilayer topology for the network, then studied the dynamics of the polarization

as it relates to interlayer couplings. They essentially operate from the assumption that the system is initially polarized, with the layers representing different opinions on a particular topic. The study focuses on three scenarios: unidirectional coupling, symmetric coupling, and nonsymmetric coupling. As is likely already evident, this is yet another theoretical approach to studying characteristics of what might represent echo chambers using synthetic data, with a bit more complexity literally layered on the previous model. The authors do strive to draw parallels between each scenario and the real world, but much work remains to close the distance between the two if we are to rely on the simulation as a viable predictor or indicator of existent echo chambers.

This same trend continues even among agent-based models of echo chambers—the simulations are based on synthetic data which then are compared and contrasted to real data, such as in [83]–[86]. Efforts to incorporate real data into echo chamber simulations are rare indeed, if existing at all.

Having examined some background topics and research conducted by others, we will now reflect on prior work in which we have been involved, noting how those efforts affected our outlook and influenced our research interests.

3. Steps Toward Fake News and Echo Chambers

Much work was accomplished in researching other projects which inspired and fed our interest in addressing the problem of fake news. Several publications reflect that work, specifically [87]–[92]. In the next section, we will review those that had the most direct and notable impacts on the direction of this research.

3.1. Insights

The research associated with "Intent-Driven Behavioral Modeling during Cross-Border Epidemics" [92] and "Modeling Emergent Border-Crossing Behaviors during Pandemics" [90] was particularly influential to the direction of our research. In [92], we employed dynamic, multi-domain modeling to explain the decisions and actions taken by actors in a scenario. We validated our approach by modeling and analyzing migration behaviors during the 2009 H1N1 pandemic in Mexico. This research fueled our interest in modeling human behavior, while the pandemic theme was relevant, as modeling the spread of information and opinions has often been related to disease-spread dynamics and vice versa [93]–[96].

Likewise, efforts towards [90] were formative, as the research employed a novel intent-driven modeling paradigm for real-world scenarios by causally mapping beliefs, goals, and actions of individuals and groups to overall behavior. We validated this approach by examining emergent behavior occurring near a national border during pandemics, specifically the 2009 H1N1 pandemic in Mexico. We accomplished this through representing the dynamism of the complex situation at multiple scales by including both coarse-grained (events at the national level) and fine-grained (events at two separate border locations) information. Such experience and insights were invaluable to addressing the

multifaceted problem of fake news in social media as that problem surfaced in American and global society.

Furthermore, the social network material explored in “Infusing Social Networks with Culture” [91] was highly relevant to the topic of social media in general. For this study, we systematically represented cultural influences in the form of relevant factors and relationships, while leveraging relevant social theories, and then infused them into social networks to obtain more realistic and complete analyses. Through highlighting the dynamics involved in complex networks, our understanding of the interplay of forces within the same was greatly improved. It is apparent that this effort was influential when considering the effects of social media on the emerging problem popularly known as fake news.

Beyond those influences, the research conducted to produce “Modeling Complex Social Scenarios Using Culturally Infused Social Networks” [97] had its own unique effect on our focus. This study explored the formation and dissolution of coalitions and groups in the face of competing and conflicting opinions, both inside and outside those groups, in the specific context of the rise and fall of the Islamic Court Union (ICU) in the 2006 Somali conflict. In a similar context, [88] explored the ability for a community to remain resilient against dramatic changes, such as those encountered by fishing communities in Somalia during the peak of Somalian piracy from 1999 to 2012. Both works contributed directly to the formation of thoughts and theories related to modeling the echo chambers to be discussed later in this work.

3.2. Explorations

As our insights developed into researching fake news detection and deterrence, we explored some topics of interest that also fueled our interest in addressing fake news.

3.2.1. Behavioral Strategies

The prevailing model in economics based on game theory relies on humans behaving as rational actors, yet this paradigm often fails to predict *actual* behavior, both in games and in real-life. People are neither perfectly nor equally rational; the same is true of all other traits. This research was spawned to close the void between perfectly rational predictors such as that produced by game theory versus actual human behaviors driven by the complex interplay of reasoning, learned behaviors, character traits, and emotions.

Our research aimed to improve modeling of decision-making in competitive scenarios by using Bayesian Knowledge Bases (BKBs) [98] to account for individual characteristics in strategies employed and resultant outcomes. BKBs are founded on Bayes' theorem, which allows probabilities to be updated as new or updated information is obtained. BKBs unify "if-then" style rule formation with probability theory, providing the ability to reason over complex situations and present probabilistic outcomes even in the face of incompleteness. Additional background on BKB use is provided in Appendix A.

Initially, we reviewed numerous studies in human behavior and game theory, searching for suitable data with which to explore the aforementioned topics. We also began with investigating options for representing cultural information about players in BKBs to improve our understanding and forecasting of the logic and influences affecting the decisions of those players in game theory scenarios. These initial forays into BKB

representations of player cultural backgrounds, traits, and strategies fed directly into the following experiments.

The first study we undertook involved data obtained from Andreoni and Miller in [99], where the authors engaged 176 subjects in a modified Dictator Game. The classic Dictator Game reported in [100] asked subjects to divide a set sum of money between themselves and an anonymous second player in a one-shot game, with the second player having no opportunity to rebut or address the decision of the first player. The modified Dictator Game employed by Andreoni and Miller differed in that the subjects played a series of games and the payoffs varied from game to game. Subjects were given tokens rather than actual money, and the consequences of sharing tokens varied from game to game, where sometimes a token held would be worth more than a token shared and vice versa.

Initially, the expectation was to apply BKBs to develop models for each player based on the sequence of decisions made in the series of games, as linked to three broad play strategies, those being selfish, optimizing, and egalitarian. With the selfish strategy, players would always keep (nearly) all the tokens, regardless of the value to the other player. Optimizers would give (nearly) all tokens to whomever would gain the most for each token, while the egalitarian strategy dictates that players would divide the tokens so the payout each player received would be equal. The BKB model is updated after each game to "learn" the style of that player, details of which are provided in Appendix A. For this study, we identified six different prediction methods to consider, as outlined in Table 3-1.

Table 3-1: Dictator game prediction methods

Random play	Randomly choose a number of tokens to keep and to share, between zero and total tokens allotted
Random strategy	Randomly choose between the three strategies identified: selfish, optimizer, and egalitarian
Last strategy	Repeat the last strategy identified for this player. For first, "guess" play is exactly half of tokens allotted
Probable strategy	Apply most probable strategy as identified by a BKB initially formed based on average player strategy distribution, but updated with each round for each individual player
Probable play	Apply a weighted move based on probability of each strategy per same BKB as for probable strategy
Max strategy	Use the most frequently used strategy by that player up until that game

While our results from this modeling at first appeared significant, with the BKB-based predictions from probable strategy and probable play having the first- and second-least normalized mean absolute error (NMAE), further examination of the data revealed a critical flaw with the setup. It was belatedly discovered that the actual sequence of the game turns was not preserved with the data, so the approach for sequentially learning the play style of each player was invalidated. In response, multiple attempts were made to complete related analysis that did not rely on the sequence of play. Table 3-2 shows the results of this effort.

Row 1 reflects the results with the original, unaltered data from [99]. As mentioned, the probable strategy and probable play prediction methods—both based on training a BKB player beginning with the probability of each strategy profile being equal to the percentage of players exhibiting that profile—performed first- and second-best, respectively. Noting

Table 3-2: NMAE for various player game orderings

	Ordering	Random Play	Random Strategy	Probable Strategy	Probable Play	Last Strategy	Max Strategy
1	w/o opt order	(6) 0.392	(5) 0.307	**(1)** 0.177	**(2)** 0.185	**(3)** 0.188	(4) 0.190
2	with opt order	(6) 0.388	(5) 0.296	**(3)** 0.170	(4) 0.181	**(1)** 0.108	**(2)** 0.143
3	all permutations	(6) 0.329	(5) 0.278	**(1)** 0.177	**(2)** 0.186	(4) 0.192	**(3)** 0.191
4	perms w/egal	(6) 0.329	(5) 0.278	**(3)** 0.202	(4) 0.219	**(2)** 0.192	**(1)** 0.191
5	perms w/opt	(6) 0.329	(5) 0.278	**(3)** 0.193	(4) 0.204	**(2)** 0.192	**(1)** 0.191
6	perms w/even	(6) 0.329	(5) 0.278	**(3)** 0.198	(4) 0.214	**(2)** 0.192	**(1)** 0.191

that the game orderings for each player in the original data did not necessarily reflect the actual order of the games presented to each player, we then applied the analysis in row 2, where all the orderings found in the original data were tried with each player's results, then the optimal ordering was selected for each individual player, where optimality was determined by matching the least error for the last strategy prediction method. Predictably, the last strategy prediction method performed best, followed by the max strategy prediction, then probable strategy and probable play. It is logical that max strategy would outperform the BKB-based predictions, as the most-used strategy for each player would naturally frequently match the last strategy used by that player. While it is reassuring that the BKB-based predictions continue to outperform random strategies, this analysis is not otherwise particularly informative.

In row 3 of Table 3-2, we computed the average NMAE obtained by applying each strategy in turn to every possible permutation of the game order. Interestingly, the two

BKB-based prediction methods performed first- and second-best for all possible permutations, which demonstrates the power of the BKB to learn patterns of behavior, even when all but one of those patterns would not match the ordering of games the player saw. Note that for rows 4, 5, and 6, the NMAE only changes for the BKB-based predictions because the only change from row 3 is that the bootstrap BKB for each player, rather than being based on player strategy distribution, is instead 100% probable to be egalitarian, 100% probable to be optimizing, or equally probable of being selfish, optimizing, and egalitarian, respectively for rows 4 – 6. Here, it is interesting to note that the BKB-based predictors lose their first and second place performance once the BKB is no longer based on the distribution of player profiles.

We learned a great deal about modeling individuals and their decision-making approaches with this study, but unfortunately were limited in the conclusions we could draw about our success due to the unknown ordering of games presented to each player. Thus, perhaps the most important lesson learned was to investigate any externally obtained data as fully as possible before transitioning to experiments.

Our second study in behavioral strategies is based on data collected from an iterated prisoner's dilemma game found in [101]. A total of 167 participants played 10 rounds against anonymous partners, after which they completed a Big Five personality trait survey and Raven's progressive matrices intelligence test. Psychologists often make use of the Big Five personality traits taxonomy to group or categorize personalities, which are: Openness to experience; Conscientiousness; Extraversion; Agreeableness; and Neuroticism [102]. Raven's Progressive Matrices [103] is a popular, non-verbal test of general human intelligence and abstract reasoning. Additional demographic data (age, sex, education, etc.)

were also collected. Our goal was to employ BKBs as a tool to model the likelihoods of individual strategies and outcomes in games based on the personal characteristics of competitors.

As mentioned, subjects were required to complete ten rounds of an iterated prisoner's dilemma game. The classic prisoner's dilemma was formalized by A. W. Tucker in 1950 [104], being a prisoner-themed scenario based on a payoff structure devised by Melvin Dresher and Merrill Flood to illustrate the properties of two-person, zero-sum games. Tucker's prisoner's dilemma describes a setup where two prisoners are charged with a crime and held separately by the police. Each prisoner is informed that if they confess and the other does not, they will be given a reward of one unit, while their partner will be fined two units. If both confess, each of them will be fined one unit, but if neither confesses, both will be released with no payment nor fine. In [101], the payoff matrix appeared as in Table 3-3. To be clear, in this scenario, if both prisoners cooperate (do not confess), they receive $1 each, if only one cooperates, the cooperator receives nothing while the other receives $1.50, and if both defect (confess), they both receive $0.25. Since both players know the payoff schedule but are not allowed to communicate with one another, they can both choose to cooperate for a modest payoff, but if one tries defecting in hopes of maximizing their payoff at the expense of the other player, it can backfire with both players receiving a mere $0.25 payoff.

Table 3-3: Prisoner's Dilemma Payoff

	Cooperate	Defect
Cooperate	$1, $1	$0, $1.50
Defect	$1.50, $0	$0.25, $0.25

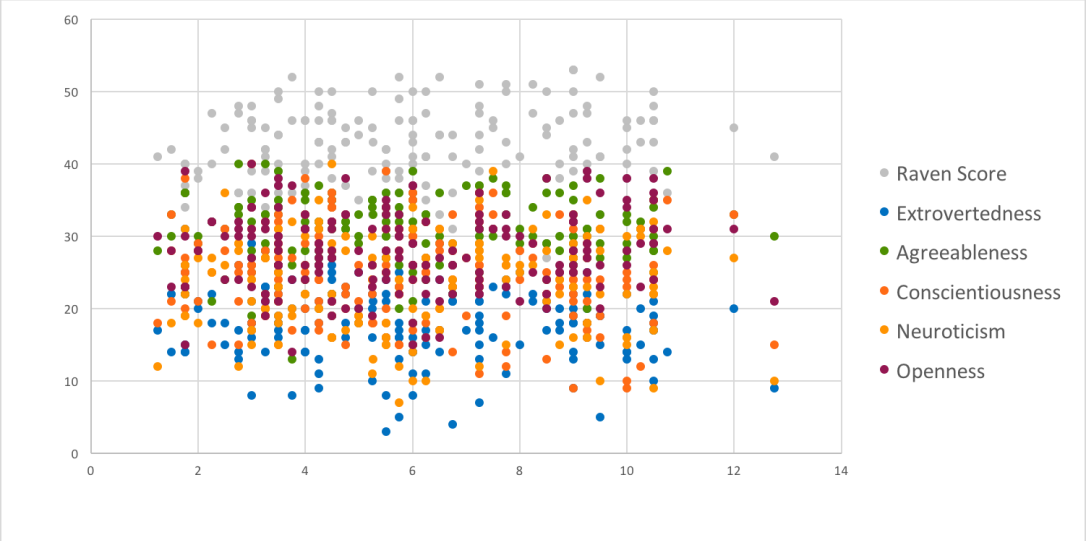

Figure 3-1: Individual payoff vs individual traits

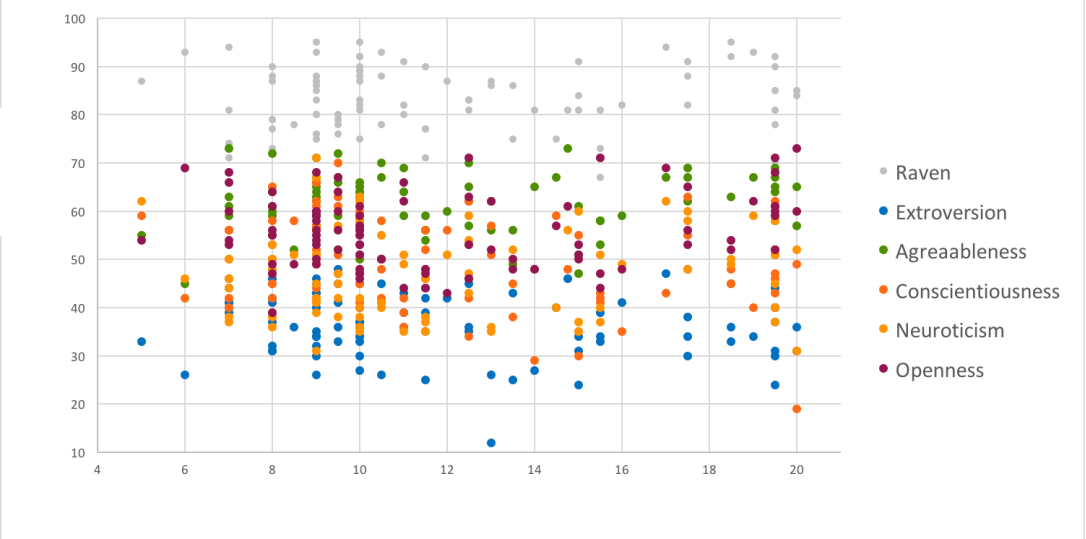

Figure 3-2: Summed paired player payoff vs. summed paired player traits

The planned direction for this study was to first analyze the payoffs of the games with traditional statistical analysis, followed by employing BKB models of the players to mimic player strategies and moves. Analysis from the perspective of individual and paired player traits yielded no insights into the prediction of player payoffs. The results of our individual player analysis can be found in Figure 3-1, while our paired player payoff analysis can be

found in Figure 3-2. These results, along with the original authors' study, convinced us that straightforward statistical analysis was unlikely to yield meaningful conclusions beyond those already published. Our hypothesis was, however, that further modeling using BKBs to investigate individual game strategies would prove more insightful. Unfortunately, the planned BKB models were never produced as this study was discontinued at this point. The intern assisting on this research completed their internship and the PhD candidate leading the research transitioned from full-time study to "in absentia" status, so only one focus for study was possible going forward, which is where the research into fake news comes into the picture. Despite the limited results from each of these studies, these forays into behavioral strategy were not without their rewards. It is hoped we may return to the topic in future. Most importantly, these efforts whetted our interest in better understanding human motivations, decisions, and outcomes—topics essential to the study of fake news.

3.2.2. Corporate Psychopathy

Concurrent with the investigations into behavioral strategies, we also examined the phenomenon of psychopathy and its occurrence in the corporate world. To start with, we explored definitions of sociopath, psychopath, and corporate versions of each. The results of this literature review [105]–[114] are summarized in Table 3-4.

Based on this review, for the purposes of this research, we defined a corporate psychopath as "a successful psychopath within an organization," where a psychopath is, in layman's terms, "a person having an egocentric and antisocial personality marked by a lack of remorse for one's actions, an absence of empathy for others, and often criminal tendencies." [115] While our survey revealed this to be an area of interest and concern for researchers, with related articles found in Australia, Canada, Great Britain, Israel, Turkey,

Table 3-4: Corporate psychopathy definitions

Source	Definition
[105]	Individuals in the workplace whom coworkers have described as "creative, strategic, having good communication skills, low management skills, poor team member, appraising lower performance, self-serving, opportunistic, manipulative, ruthless, shameless, charming, grandiose, and ambitious."
[106]	Psychopathy in the workplace where likely the individual's psychopathic traits relate more to good impression management as opposed to good job performance.
[107]	A psychopathic individual who works for an organization but is more concerned with personal success and self-enrichment instead of that of the organization for which they work.
[108]	Someone with a lack of conscience, a key feature of psychopathy*, who manages to avoid legal confrontation with authorities and works for an organization.
[109]	A psychopath who successfully evades the attention of the authorities and works within an organization.
[110]	An individual who presents manifestation of psychopathic traits, who has not been incarcerated in the judicial or mental health systems, and is more likely to engage in manipulative and antisocial behavior and works in a corporate setting.
[111]	A dysfunctional leader with psychopathic traits, which are likely underlying factors in their deviant interpersonal behaviors.
[112]	Someone who displays sociopathic characteristics and works for an organization.
[113]	Someone who works in the business sector and exhibits psychopathic characteristics. These people have implications in counterproductive workplace behaviors, white collar crime, ethical decision-making in the corporate world, and leadership.
[114]	An employee, often a leader, who demonstrates aggressive behavior in the workplace. A person who manages employees and manifests deviant behavior characterized by bullying, using violence, and performing passive or active acts of aggression.

and the United States, such studies tend to rely on self-reporting of characteristics to provide input to the assessment of the existence of psychopathy, and the participation from corporations, and particularly corporate leaders, to allow for such assessments is rare. Thus, there exists a paucity of reliable data to allow effective analysis of corporate psychopathy.

To overcome this lack of data, we focused on trying to understand the patterns and origins of psychopathic corporations, then appled game theoretics to explore implications of the existence of corporate psychopathy. Our initial foray into this realm began with modeling the emotional gameplay of participants in a voluntary prisoner's dilemma game. Wang et al. [116] posited that players imitate emotional profiles of opponents rather than imitating opponent strategies. For the voluntary prisoner's dilemma, rather than simply having the strategies of cooperate or defect, players also have a third strategy dubbed "loner", where a player could temporarily withdraw from the game but still receive a small, fixed income σ. The payoff matrix for this voluntary prisoner's dilemma is outlined in Table 3-5.

Table 3-5: Voluntary prisoner's dilemma payoff [116]

Player 2 Player 1	C	D	L
C	R/R	S/T	σ/ σ
D	T/S	P/P	σ/ σ
L	σ/ σ	σ/ σ	σ/ σ

R(reward)=1, S(sucker)=0=P(penalty), $1 \leq T(\text{temptation}) \leq 2$, and σ=0.3

The authors' simulation results are reflected in Figure 3-3, which we were able to successfully replicate (Figure 3-4). The aim of this effort was to eventually tie the emotions the researchers identified for weaker players in this game, those of sympathy, harshness, and apathy, and for stronger players, those of respect, jealousy, and fear, to emotional profiles for corporate psychopaths through the application of game theory. Future research was envisioned as applying research of corporate decision making to agent-based modeling, creating a simulation embodying corporate decisions, and looking at long-run

effects and trends.

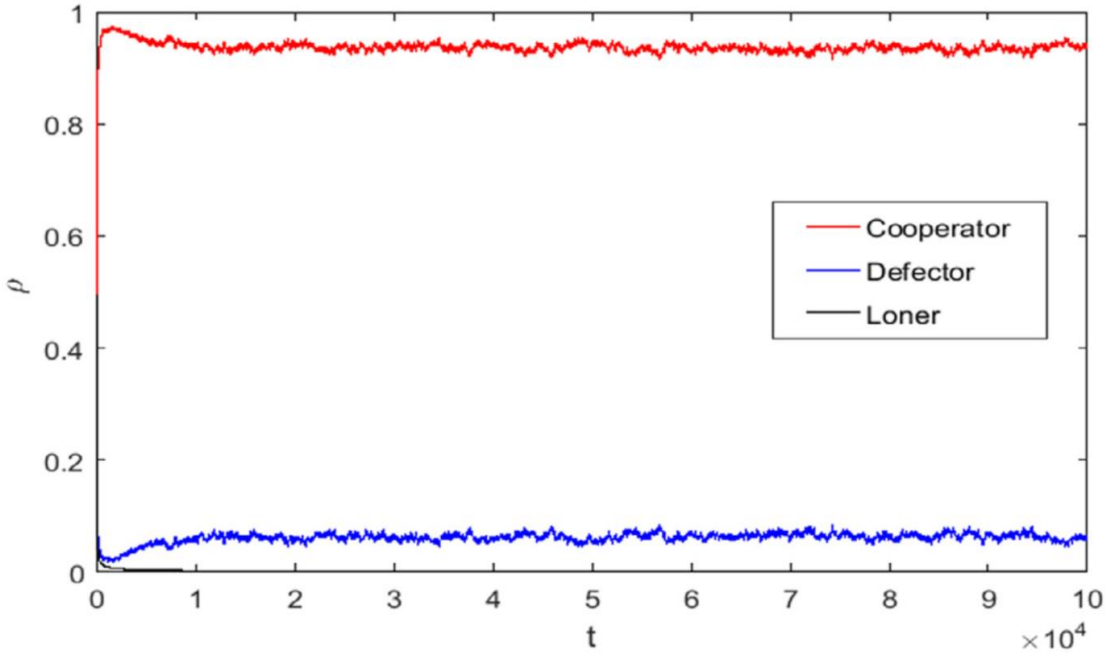

Figure 3-3: Fraction of cooperators, defectors, and loners from [116] over time

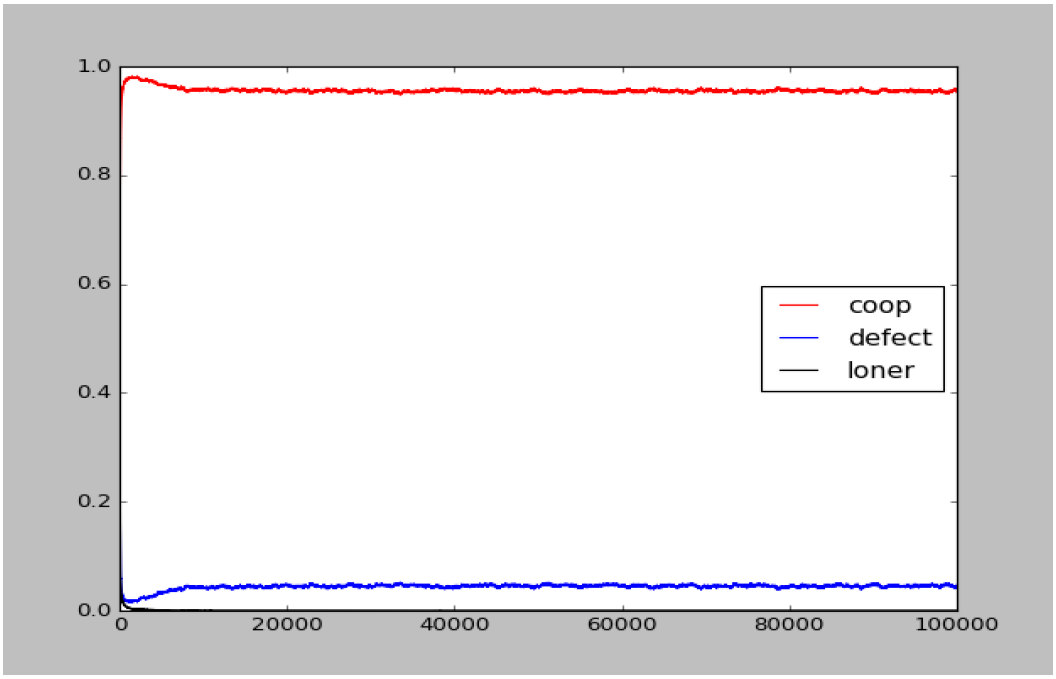

Figure 3-4: Replicated results

Unfortunately, once again this research was discontinued for several reasons. Most pressing were the limitations in studies that included senior corporate personnel which limited the ability to analyze the most influential leaders in corporate business, but also due to personnel changes and shifts in focus. Nevertheless, the insights gained into human behaviors in the corporate world would prove exceptionally relevant to our ultimate focus on fake news, which is where our research picks up in the next section.

4. Fake News Model Formation and Exploration

In order to usefully provide a method for recognizing fake news, individual biases and opinions must be considered. The labeling of fake news is a subjective effort, as the identification of "truth" in our complex world is not always straightforward. Moreover, with the increasingly polarizing effect of modern politics, what is accepted as truth and what is rejected as fake has often become more a personal issue than an evidentiary effort [117]. What one person may accept for fact, another may argue is fiction, as succinctly argued in [118]. For this reason, it is proposed that the problem of fake news be partitioned into two pieces: premises and misleading content. Those two parts then form a framework to effectively evaluate the validity of any potential news.

The value of this research and the accompanying framework lies not only in its practical utility to potential users, but also in the light it will shed on the prevalence of bias and the absence of balance in many news stories. It is conceivable it could also be applied to historical news sources, to gauge if news has truly become less objective in recent years. Furthermore, its biggest eventual contribution to science could be in plainly identifying attempts at undermining actual science with biased reporting and pseudoscience.

The foundation of this research is in identifying the components of a fake news item. While there are numerous approaches to fake news and numerous ways to try to identify or rate fake news as discussed in Sections 2.1 and 2.3, there are only a limited number of ways that the information in fake news can be automatically and reliably parsed for signs of counterfeit. This proposal contends that a fake news piece, as defined previously, can be condensed down to a collection of assertions, which may or may not be true, and various methods of deception, including biased arguments and language, misdirection, and

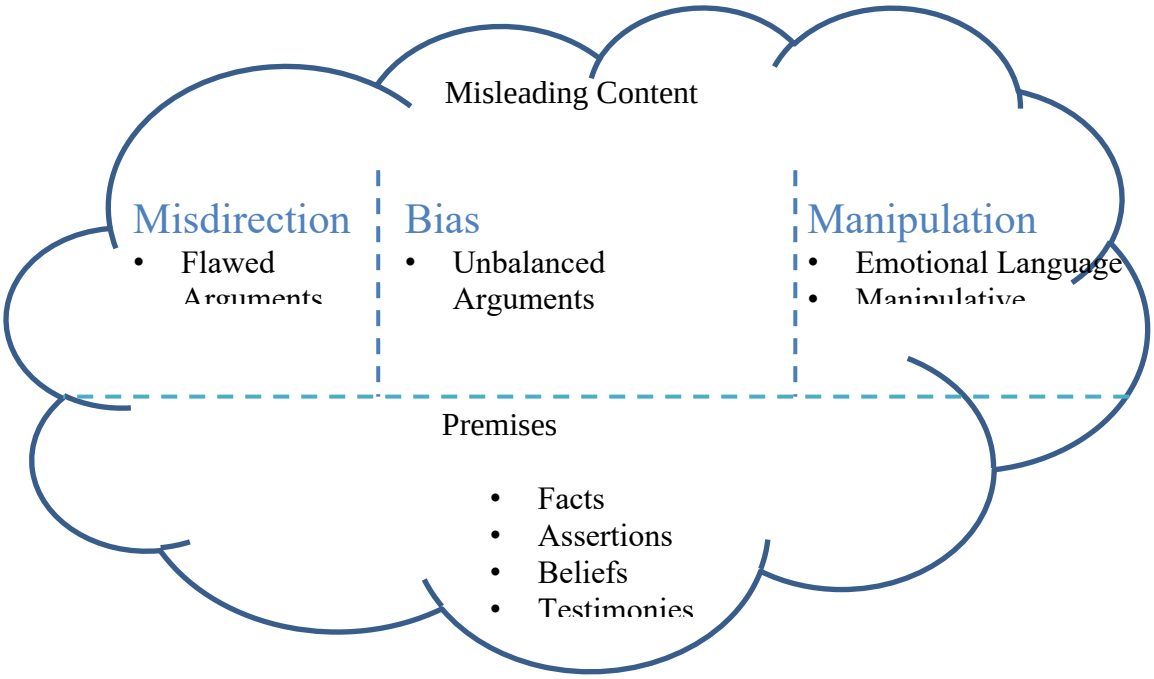

Figure 4-1: Fake News Framework

manipulation. While these components are not comprehensive, it is believed they are some of the most recognizable and most critical. Further components may be added as this research progresses. A brief description of that framework follows, and an illustration of its composition can be found in Figure 4-1.

Although the intent here is to identify discrete components, some overlap and blurring of definitions is unavoidable. For example, emotional and manipulative language may be employed to support a biased argument. Regardless, if both manipulative language and a biased argument are present, they magnify the effect of either alone. Additionally, every effort is made to ensure the framework is all-inclusive, but that is a tall order. One approach to help meet this objective is to employ the typology of fake news defined in [37].

The authors note that after a survey of 34 academic articles focused on fake news,

they found the following fake news types: satire, parody, fabrication, manipulation, propaganda, and advertising.

For the purposes of this research, the first two types, satire and parody, do not fit the chosen definition for fake news, as they do not represent themselves as genuine news. While the "fake news" sobriquet is often applied to works of satire and parody, even by the creators of said works, they are clearly provided for entertainment. Moreover, as noted in [37, p. 142], "the core content of political satires is based on actual events." The parodies are of course a different story. They emphasize "the ludicrousness of issues…by making up entirely fictitious news stories." A similar argument applies for the last type, advertising, which is clearly not labeled as news, except in the relatively rare instances when advertising attempts to pass itself off as a news item in some publications. While that does occur at times, that can be easily remedied through requiring advertising to be transparently labeled. For these reasons, advertising is not considered within the scope of this study.

The remaining three types—fabrication, manipulation, and propaganda—are central to the subject of study and will be addressed. In short, fabrication would fall within the premise category, unless there is an attempt to support the fabrication with argument, in which case any or all the other components of fake news could be employed in that effort: bias, misdirection, and manipulation. The manipulation type of fake news is obviously captured by the manipulation component identified in the fake news framework. However, the primary focus of the authors of this typology [37] was image manipulation, whereas the focus here is on manipulative language. Propaganda could employ any or all four components of fake news. These three types primarily differ from the others in their motivations—they all intend to mislead, which is of interest to this study, but not

exclusively so.

'House oversight panel votes Clinton IT chief in contempt'
'Story highlights The House Oversight panel voted 19-15 to hold Bryan Pagliano in contempt.

Pagliano failed to show for a second hearing on Clinton's emails and private server

Washington (CNN) Members of the House Oversight and Government Reform Committee voted along party lines Thursday to hold the architect of Hillary Clinton's private email server in contempt for failing to appear before them.

The panel's 19-15 vote came after Bryan Pagliano failed to attend a second hearing on Clinton's emails and use of a private server while leading the State Department. Republicans blasted his decision as an act of defiance, but Pagliano's lawyers said the demand that he appear "betrays a naked political agenda."

House Oversight Chairman Jason Chaffetz, a Utah Republican, argued that previous testimony from another one of Clinton's IT workers, Justin Cooper, made it imperative for Pagliano to testify.

"I explained Mr. Pagliano was uniquely qualified to provide testimony to help the committee better understand Secretary Clinton's use of a private email server. This is indisputable," Chaffetz said. "I also made clear the committee would consider all options regarding Mr. Pagliano's failure to appear, including consideration of recommending he be held in contempt."

Figure 4-2 : Sample Real News – Common Definition

As a demonstration for discussion, Figure 4-2[9], Figure 4-12[10], and Figure 4-4[11] contain sample news articles from a fake news analysis database [39], [119], which will be

[9] "House oversight panel votes Clinton IT chief in contempt" cited at 'http://cnn.it/2deaH2d'.

[10] "NY Gov. Andrew Cuomo disputes Trump's claim cops are 'afraid' to do their jobs" cited at 'http://cnn.it/2d3tTeL'.

[11] "The DEA Just Raided A United States Senator–Dems In A Panic – News Feed Hunter" cited at 'http://newsfeedhunter.com/the-dea-just-raided-a-united-states-senator-dems-in-a-panic/'.

'**Governor defends providing medical care for bombing suspect** '
Story highlights "I don't know how you could have been more aggressive than we were here," Cuomo said

Ahmad Khan Rahami was captured Monday after a manhunt and shootout

Washington (CNN) New York Gov. Andrew Cuomo said Tuesday that Republican presidential candidate Donald Trump is wrong to suggest that the New York bombing suspect shouldn't receive medical care.

"I understand the anger that Donald Trump is speaking to (but) this is America and this is our system and you are innocent until proven guilty and you have a right to counsel and that is the Constitution of the United States of America," he said on CNN's "New Day."

"And that's what makes us who we are. That's what makes us special. And if you give that up, Alisyn, then you have defeated yourself," Cuomo, a Democrat who has endorsed Hillary Clinton for president this cycle, told CNN's Alisyn Camerota. "That is the code of democracy and freedom. That is what they resent about us. So don't lose your soul in the process. Because that is the soul of America."

Trump complained Monday about Ahmad Khan Rahami being offered medical treatment and legal assistance following his shootout with police. Rahami, the suspect in Saturday's bombings in New York and New Jersey, was captured Monday. Following a frantic manhunt and shootout, Rahami was shot multiple times before being taken to a hospital for surgery.

"He will be taken care of by some of the best doctors in the world. He will be given a fully modern and updated hospital room. And he will probably even have room service, knowing the way our country is. And on top of all of that, he will be represented by an outstanding lawyer," Trump said at a Florida rally.

Figure 4-3: Sample Real News – Framework Definition

referenced in the following subsections. These citations were chosen specifically for their similar lengths but dissimilar real-news contents. Note that the citation in Figure 4-2 was assessed by the fact-checking website PolitiFact[12] to be a fact-based real news story, as was the citation in Figure 4-12. While both are considered real news according to the source repository, we will highlight their distinctness later. The citation in Figure 4-4 was assessed

[12] https://www.politifact.com/

to be a fake news story. Figure 4-5, Figure 4-6, and Figure 4-7[13] contain example argument structures for those citations, which will also be referenced in the forthcoming subsections. In the diagrams, a red box indicates a refutation to the linked argument, and all subordinate boxes to the red box are supports for the refutation.

'The DEA Just Raided A United States Senator–Dems In A Panic – News Feed Hunter'

'The DEA just raided the vacation ranch of Democrat Senator Hal Lindsay (D-NJ), seizing more than 400 marijuana plants, 2 greenhouses full of opium-producing poppies and a small lab that was pumping out massive amounts of refined, finished product. The ranch, just a few hundred miles north of anything in Wyoming, was also seized along with a fleet of automobiles, recreational vehicles and other property now considered the spoils of the drug trade.

The bust itself yielded more than $6 million in finished drugs alone, never mind the plants and raw product waiting for packaging. All in all, Senator Lindsay is looking at 70 years behind bars on the opium alone. He was taken into custody at his office in Washington DC and has since been booked and released on $10 million bail.

Lindsay's office isn't commenting on the ordeal but New Jersey Governor Chris Christie has already called for his removal and a special election as soon as possible. The loss of yet another seat, especially in the northeast, would be devastating to the Democrats.

The DEA says their investigation isn't over and that there are potentially more politicians involved. We'll keep you updated.'

Figure 4-4: Sample Fake News

[13] Argument structures produced using Araucaria, found at http://araucaria.computing.dundee.ac.uk/doku.php or https://www.softpedia.com/get/Others/Home-Education/Araucaria.shtml.

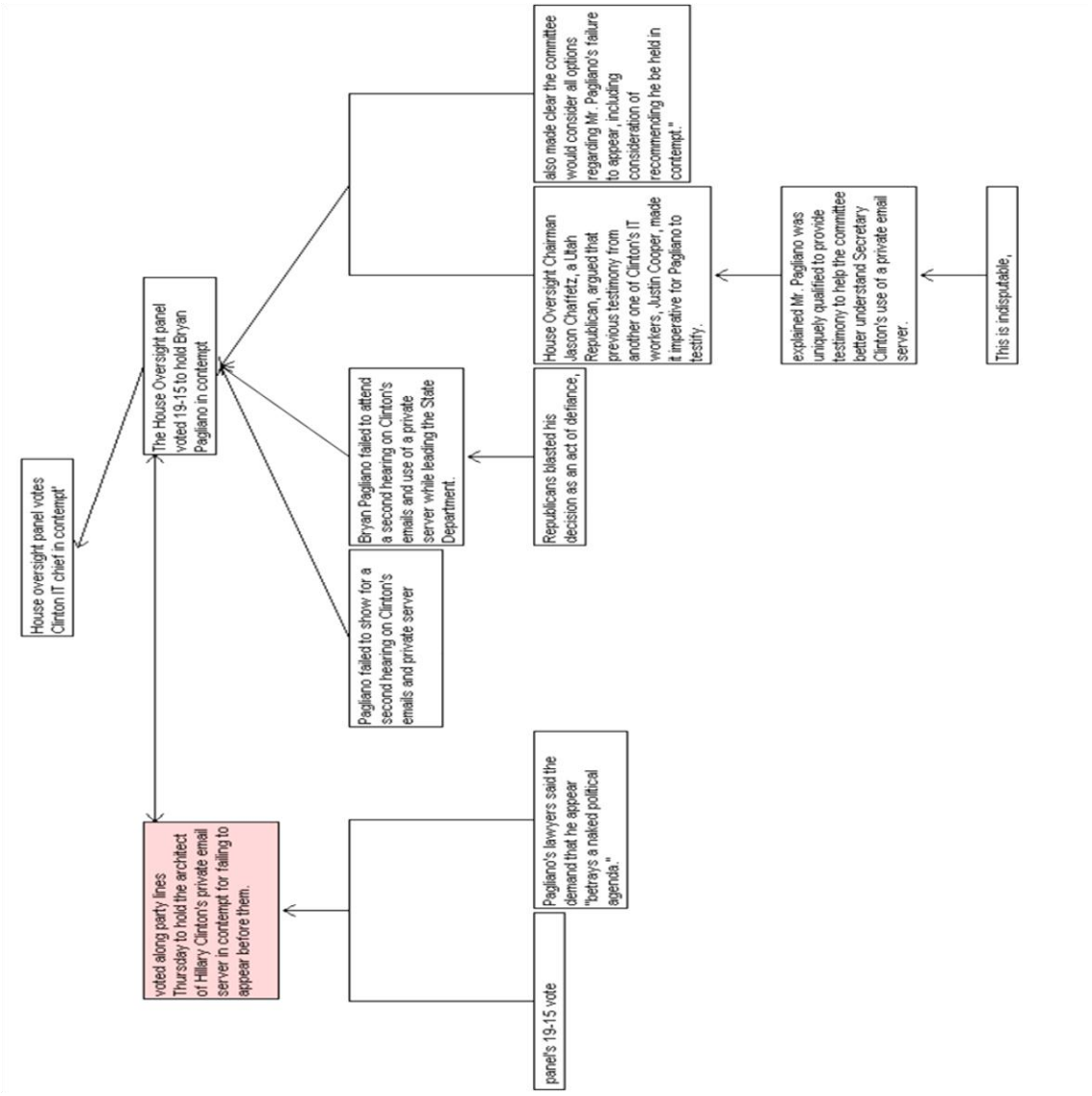

Figure 4-5: Unbalanced Argument for Real News Citation in Figure 4-2

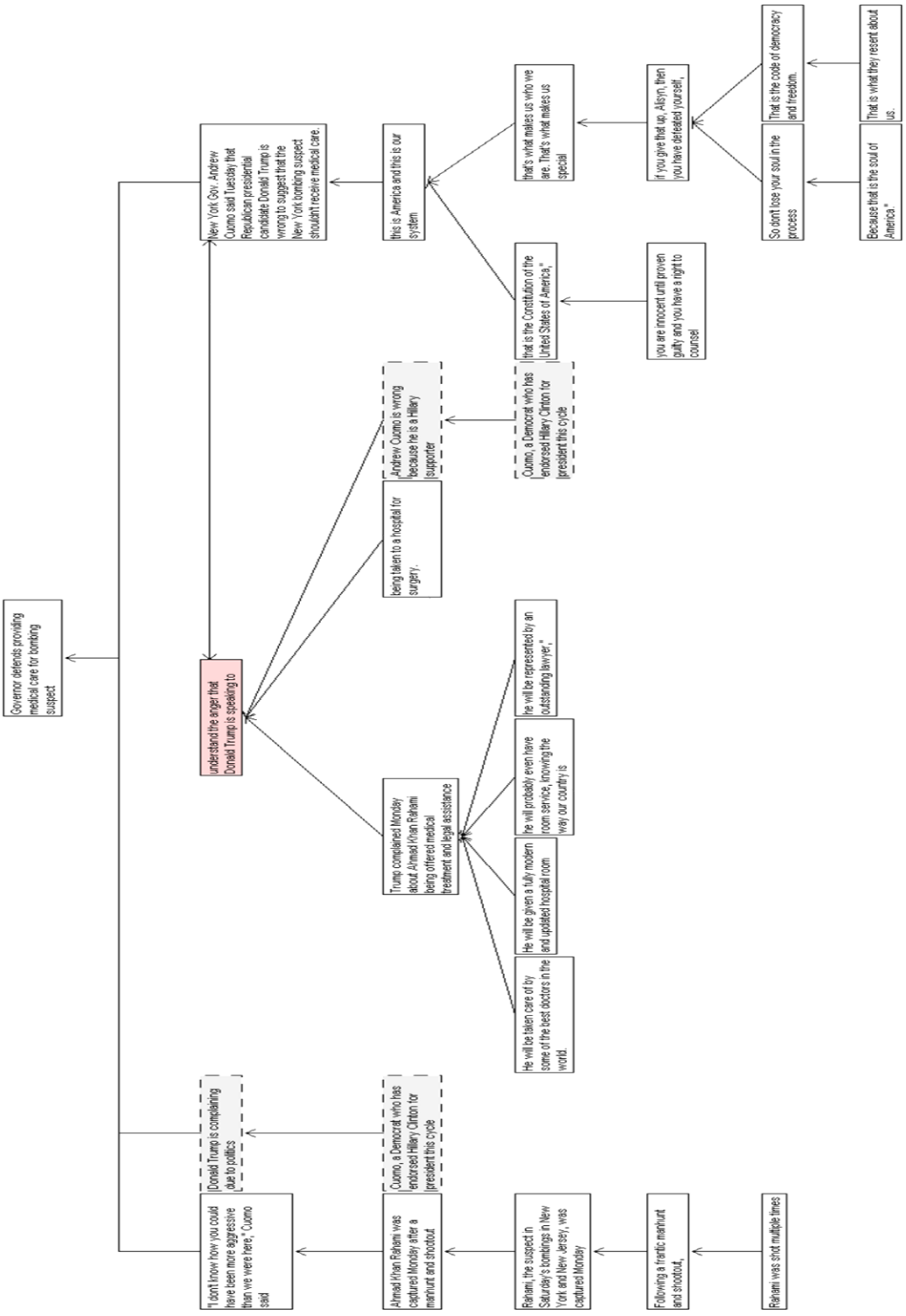

Figure 4-6: More-Balanced Argument for Real News in Figure 4-3

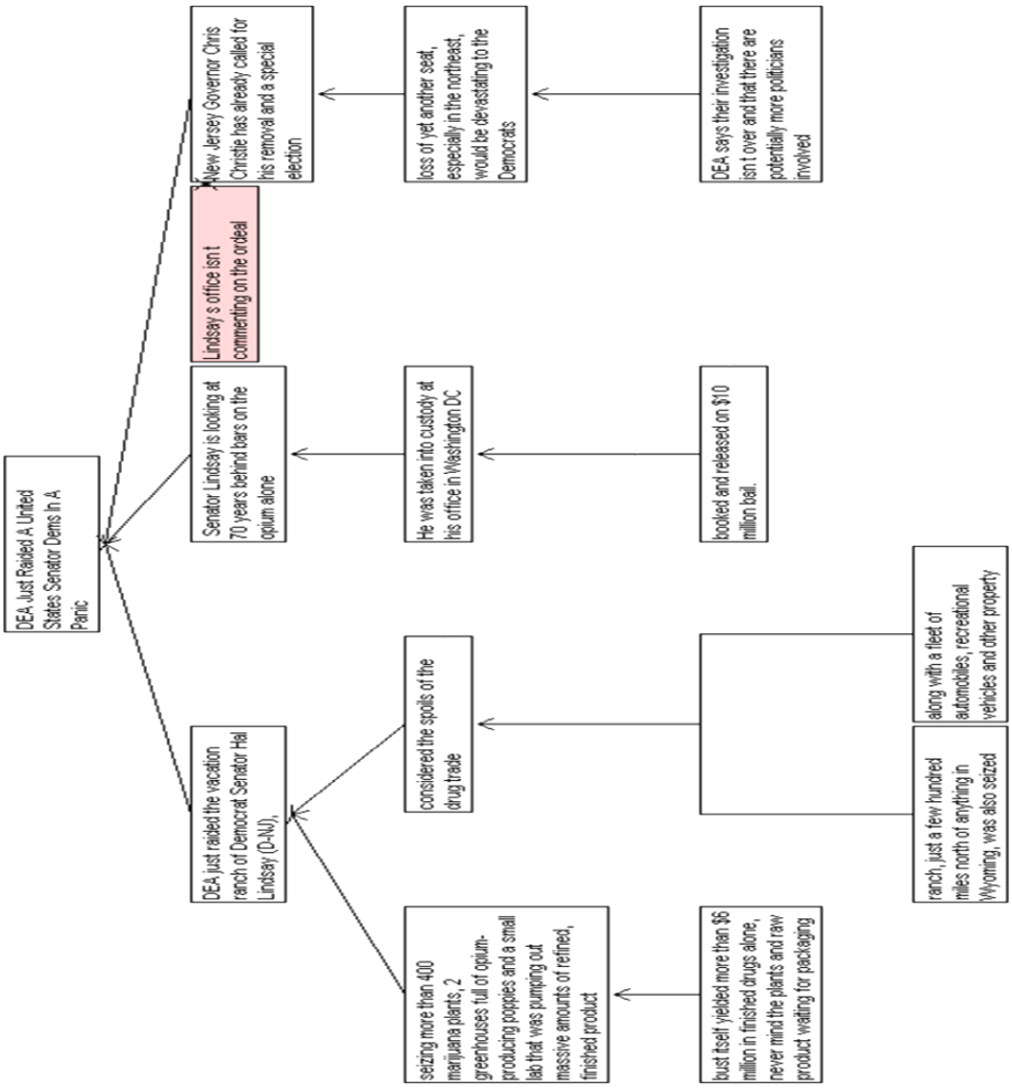

Figure 4-7: Unbalanced Argument for Fake News in Figure 4-4

To continue with this section, we introduce the concept of document graphs, which opens the door to argument construction and analysis. We then examine the potential for premise recognition, which also aids in argument analysis. Finally, we review options for evaluating misleading content in fake news before discussing conclusions from our proposed fake news model.

4.1. Document Graph Analysis

Document graphs (DGs) are a method devised to analyze text structures and employed for text summarization [120], [121]. A DG is a directed acyclic graph with nodes representing the concepts or entities contained in a document, and edges representing the relationship between those nodes. The application of DGs was explored in this research because it does not rely on the vocabulary used within documents, but instead explores the structure of the language used. This is valuable because once the specific language is introduced into the analysis, particularly with machine learning and neural network approaches, it is difficult to generalize across domains. Many of the language cues selected by automatic classifiers tend to be topic-specific and thus not useful for fake news recognition for unrestricted news subjects.

The DG is automatically constructed by parsing a document for noun phrases, which become the nodes in the DG, and then labeling the edges between the nodes with one of two relationships, either "is a" or "related to". This in essence provides a spatial element to visualizing the structure of a text document. In this research, the Stanford parser[14] [122] was employed to parse for noun phrases. Subsequently, the noun phrases are processed to generate the relations between and within the noun phrases, as described in [120]. For example, the simple noun phrase "State Department" generates the relations "State – related to – State Department" and "State Department – is a – Department." Figure 4-8,

[14] https://nlp.stanford.edu/software/lex-parser.shtml version 3.6.0

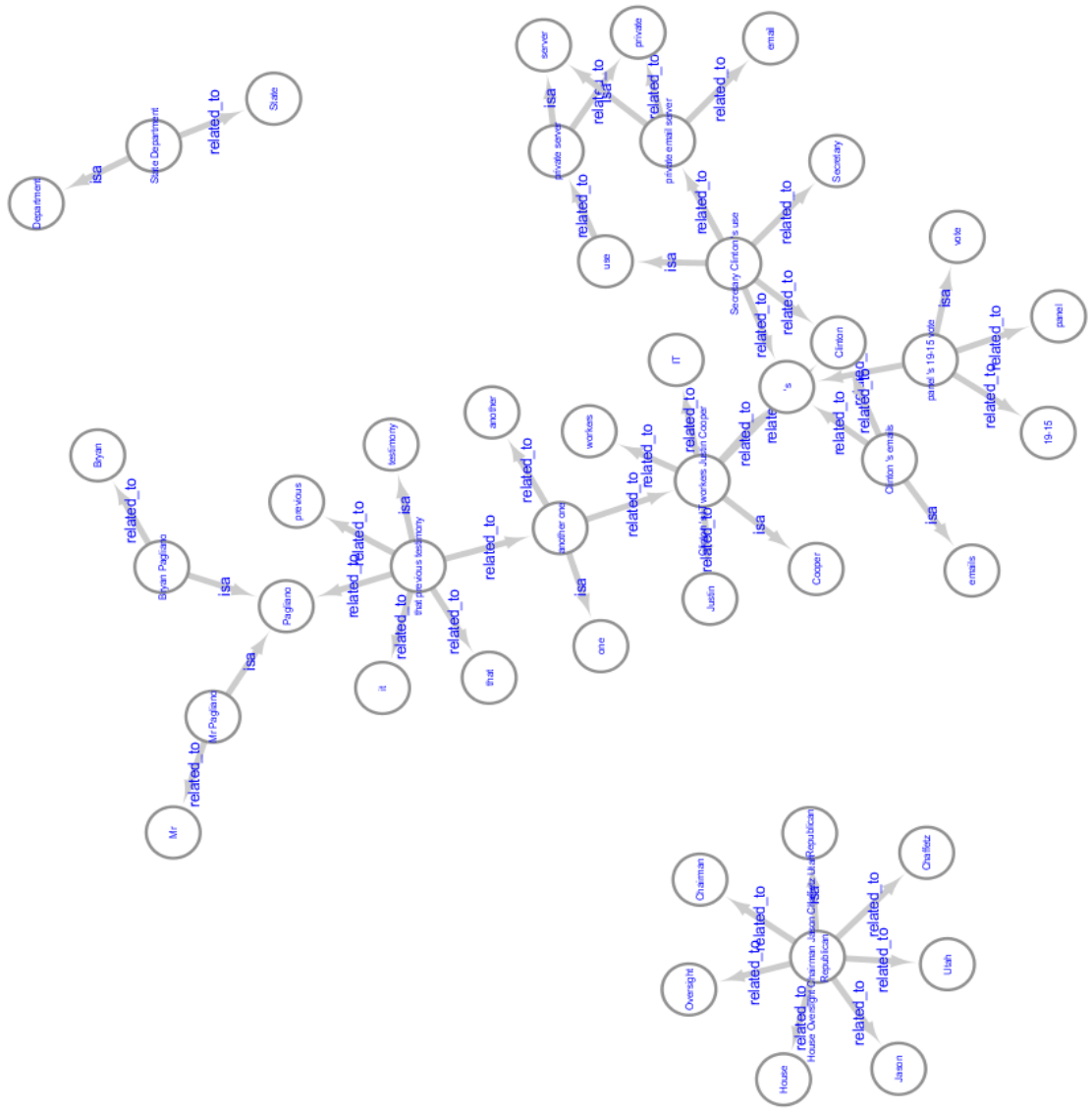

Figure 4-8: Sample Real News Document Graph

Figure 4-9, and Figure 4-10 contain the DGs generated from the news samples in Figure 4-2, Figure 4-3, and Figure 4-4, respectively. These citations were in fact chosen partly to ensure that each example had a similar length and similar DG complexity.

As evidenced in Figure 4-8, Figure 4-9, and Figure 4-10, we now have a spatial representation of the texts of interest. On the surface, these three DGs have many

similarities—they all have multiple subgraphs, some smaller, some larger and more complex. The question which we investigated was: “Are there structural components which

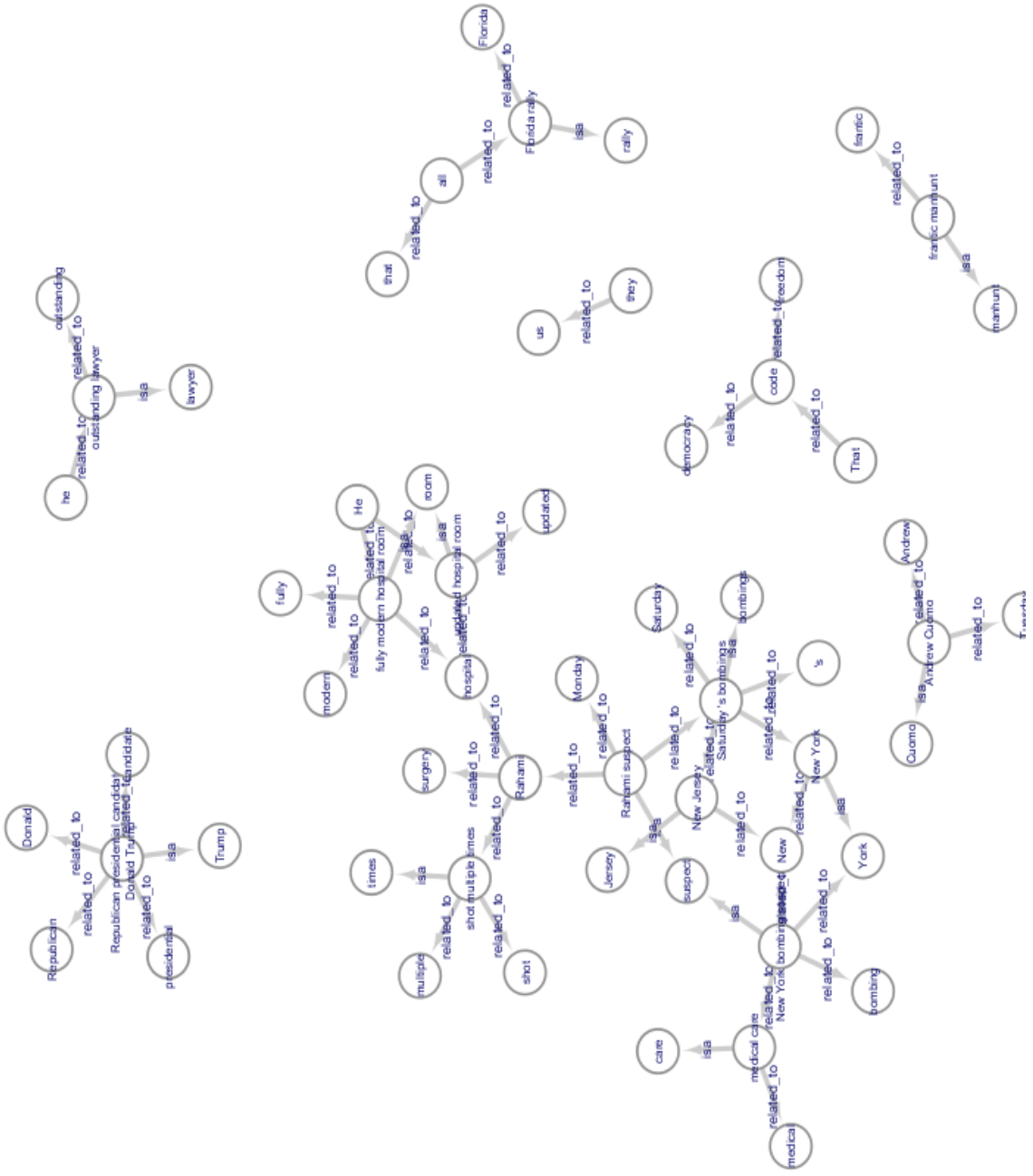

Figure 4-9: Sample More Real News Document Graph

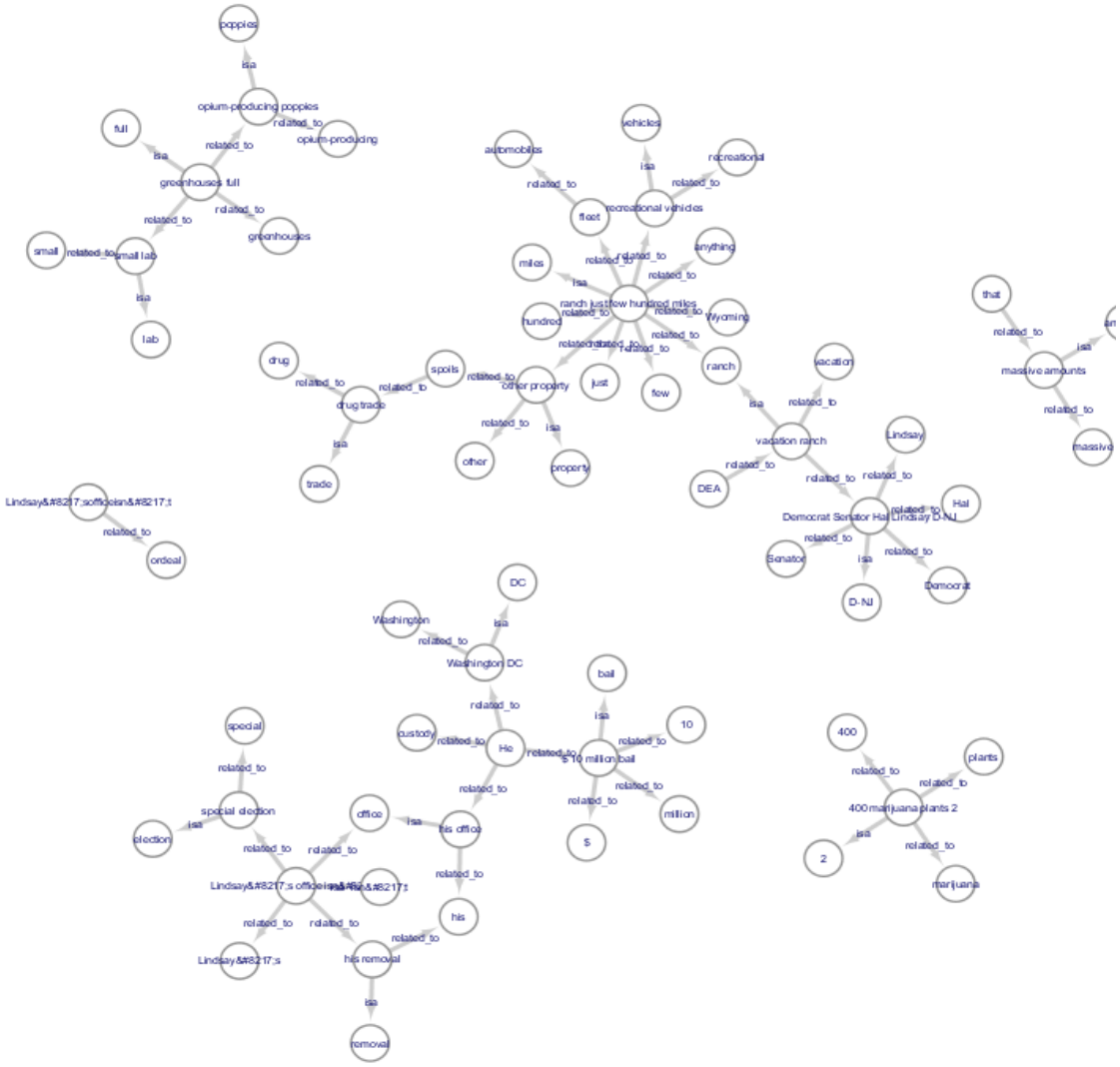

Figure 4-10: Sample Fake News Document Graph

might set real news stories apart from fake?" Examining just these three DGs is not useful in answering that question but examining certain network characteristics of the entire sets of data might. Since initially the data is only coarsely characterized as real or fake according to the criteria established by PolitiFact, that is the classification examined. The goal is to establish whether a straightforward application of DGs to the existing definitions of real and fake news can distinguish between the groups. To accomplish this, DGs were generated for the entirety of data presented in [119], including real and fake news from

both the BuzzFeed and the PolitiFact sites.

Two possible network measures that might help distinguish between fake and real DGs are betweenness centrality and closeness centrality. These two are used as a first glimpse into the utility of DGs for application to this study. They were selected to explore because they are standard measures, are well-understood, and can provide information on the overall structure of the document graphs.

4.1.1. Betweenness Centrality (BC)

BC is a measure of the centrality of the nodes in a connected graph, where the shortest path between all pairs of nodes in the graph is calculated, and then for each node, BC is calculated as the total number of shortest paths that pass through it. This number can then be normalized by the total number of nodes in the graph. BC can provide insight into which nodes are most important to a graph and give an indication of the distinctness of that importance in the graph. To make use of this metric, because the DGs are not necessarily connected, as evidenced in Figure 4-11, the betweenness centrality was aggregated for the entire DG. Two aggregations were explored: the statistical mean and maximum. Table 4-1 shows that the p-value for the difference in means of the real versus fake news BCs suggests the means are not significantly different. Furthermore, both the mean and maximum BCs do not distinguish between the real and fake news datasets. The standard deviations for both the

Table 4-1: Betweenness Centrality Compared

Betweenness Centrality	mean BC			max BC	
	mean	std	p-val	mean	std
Real News	0.00076	0.002	0.1340	0.00716	0.01297
Fake News	0.00052	0.00118		0.00558	0.00928

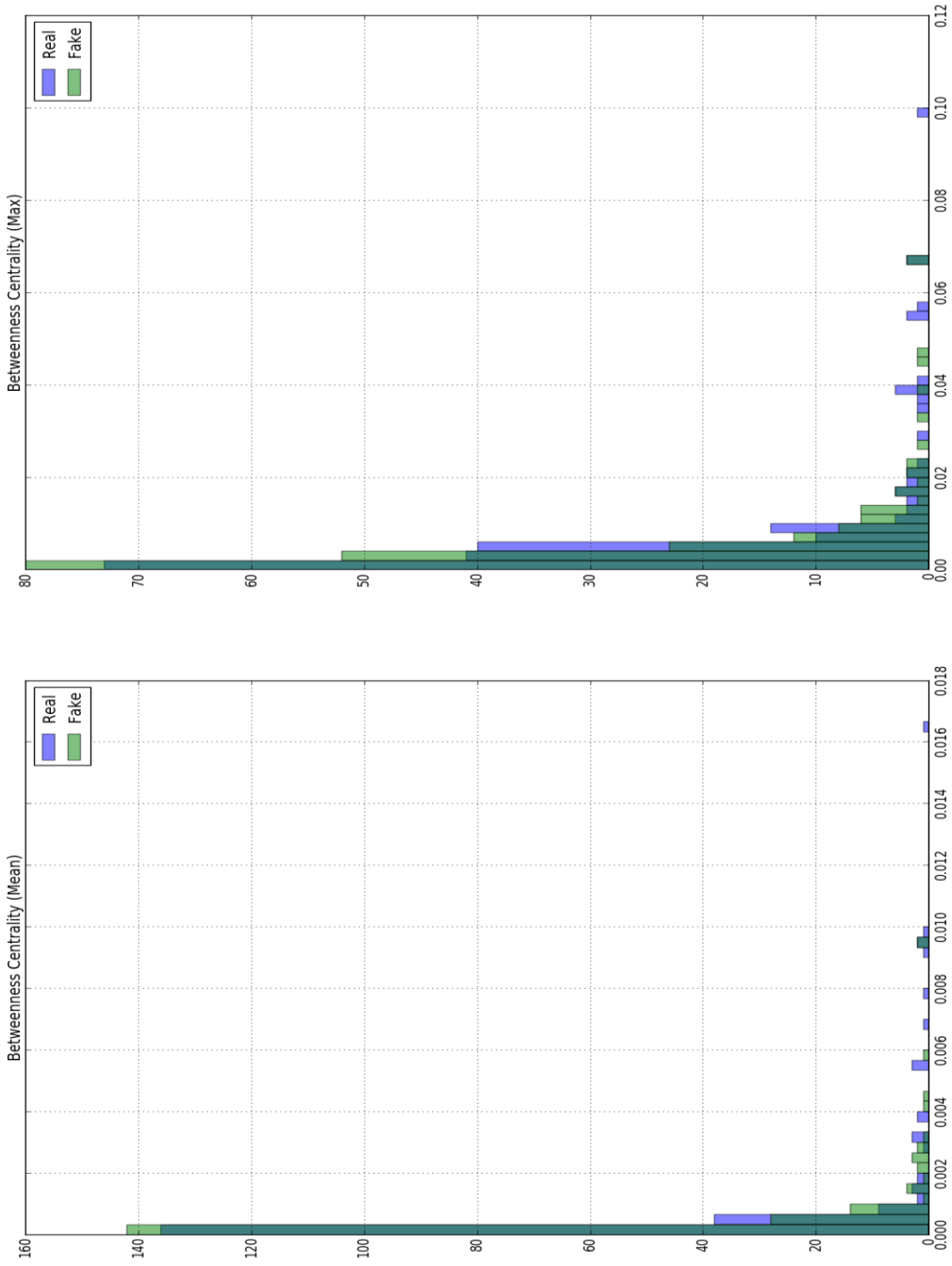

Figure 4-11: Betweenness Centrality of Real and Fake News

mean and maximum BCs completely overlap the separation between the two groups' means and maximums. This overlap or lack of separation is visibly evident in the histograms in Figure 4-11.

These results suggest that the BC of DGs for real and fake news is unlikely to provide distinguishing information for recognizing fake news. While it is possible that looking at what nodes have the maximum centrality for each DG might provide some insights, it is more likely that delving into the individual nodes, and therefore the specific noun phrases that are most central to each document, will devolve into words and phrases specific to the topic of that document, which is undesirable when trying to produce a general solution for recognizing fake news.

4.1.2. Closeness Centrality (CC)

CC is a measure of the closeness of each node to all the other nodes in a connected graph. It is calculated by taking the sum of the length of the shortest path from a node to every other node in the graph, and then inverting it. Thus, CC is greatest for the node with the smallest sum of shortest paths. Because the DGs are not necessarily connected, the mean and maximum CC for each DG will again be utilized when comparing the fake and real news. As with BC, Table 4-2 and Figure 4-12 reveal that CC does not provide sufficient separation for distinguishing between real and fake news, at least by the definition used in this dataset. The difference of means p-value reveals that the means are not significantly different. Once again, the standard deviations indicate there is not significant distinction between the means and maximums to provide differentiation.

Table 4-2: Closeness Centrality Compared

Closeness Centrality	mean CC			max CC	
	mean	std	p-val	mean	std
Real News	0.02652	0.03348	0.1803	0.15747	0.13727
Fake News	0.03269	0.05779		0.1717	0.15878

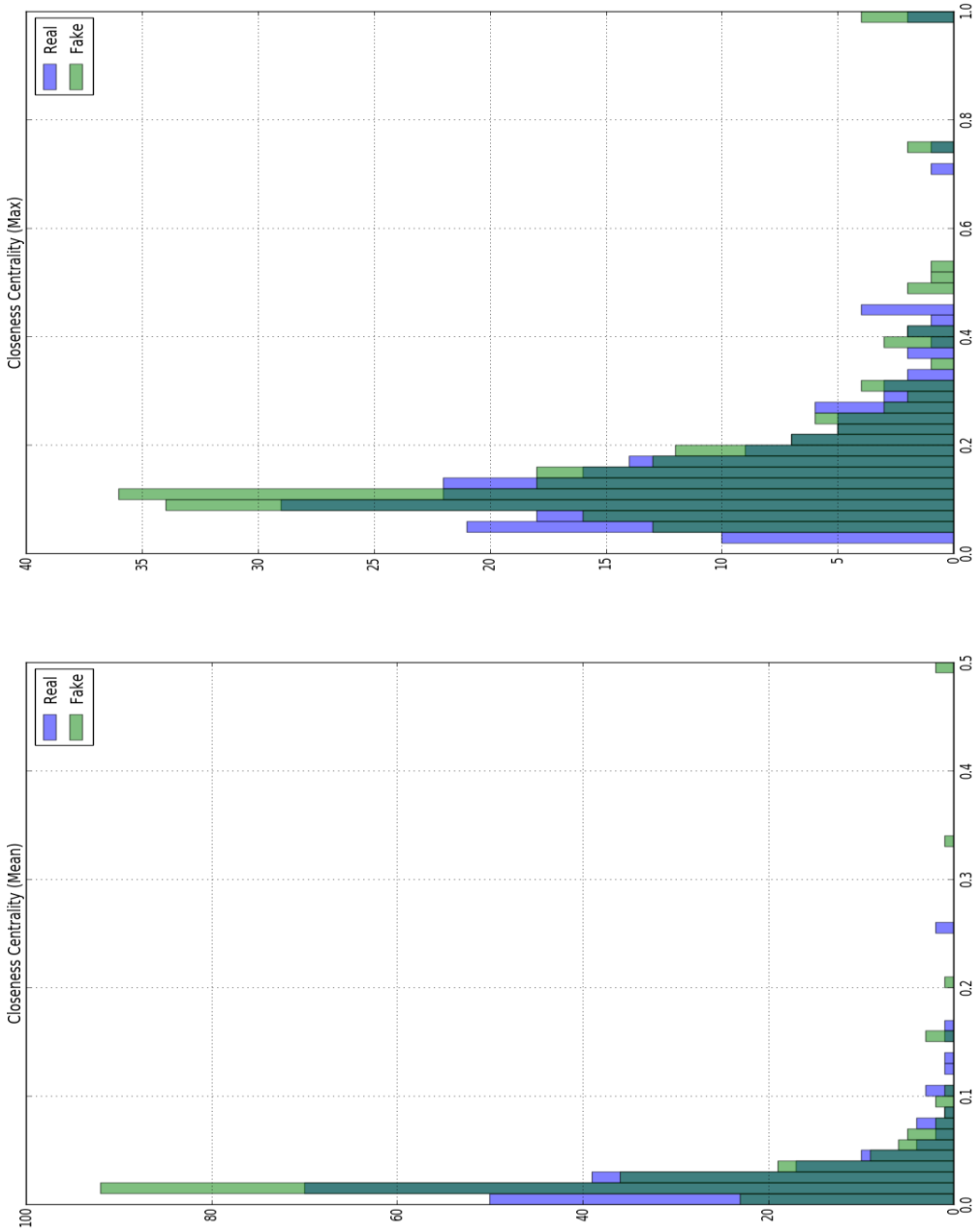

Figure 4-12: Closeness Centrality of Real and Fake News

The individual nodes could be examined for the CC measure, but this would likely only yield document-specific details not useful for a generalized discriminator for fake news. Thus, DG analysis in this small foray has not yielded significant results, but that does not

preclude the possibility that further explorations of DGs using other network techniques could be more effective. At this time, we reserve this approach for future studies.

4.2. Premise Recognition

To aid defining argument structure, an automated method to identify the premises (or claimed facts) in an article must be produced. It is suggested that using argumentation analysis, premises be identified and clearly revealed to the user. This will allow the user to know where the discussion begins, and under what assumptions. It is then up to the user to determine whether to accept those premises or whether to fact-check them in some manner. These premises could be completely true and factual, wholly false and fabricated, or range anywhere between. Interestingly, it may also be argued that the reader will have his or her own premises which could interfere with or distort the premises of the author. By identifying the premises of the article, this tendency can at least be partially controlled or averted, should the reader be motivated to genuinely identify the message of the news piece.

It is also feasible an automated or online fact-checking capability could be incorporated into the framework, but this is not essential, and could possibly introduce undesirable subjectivity. While it would appear at first glance that fact-checking would be the foundation for guarding against fake news, as will be discussed later, facts can be employed with misleading logic and manipulative language to produce an erroneous or, at the very least, false conclusion.

Another point to note at this juncture is the concept of intent. Some might consider a person’s intent is to deceive because they promote a falsity as truth. An extreme counter

example is that of the group of people who continue to contend the earth is flat [123]. While scientists and scientific-minded people firmly agree that the earth is in fact not flat, authors of flat-earth articles presumably believe their position and will passionately argue it in their publications. Measuring whether they intend to deceive or not is not productive. Likewise, measuring intent to deceive is not useful for recognizing fake news. A sincere persuasive argument based on ill-founded premises can be just as misleading and harmful as an insincere one intending to deceive.

Identifying premises will provide a foundation for the stability of any arguments presented, and then recognizing the remaining components will provide further illumination on the strength and legitimacy of the entire piece. Identification of the premises and conclusion of an argument has been described as one of the four primary tasks of argumentation [124], as is a second task labeled analysis, which is to discover implicit premises and conclusions. Clearly, premise recognition is a fundamental part of argumentation and will provide significant assistance to the fake news framework. In the sample news stories, unsupported assertions exist in all of the arguments, and are evident in Figure 4-5, Figure 4-6, and Figure 4-7. The premises are the "leaves" on the argument tree, i.e. the boxes with no arrows pointing to them.

Having viewed the potential for inclusion of argumentation in our fake news model, we will now explore methods for identifying misleading content.

4.3. Evaluation of Misleading Content

Finally, a process for recognizing misleading news must be developed, which presents the most challenging aspect of this approach to recognizing fake news. Misleading information

can take many forms and have numerous origins. The most significant approaches to misleading news, according to the framework for fake news, are bias, manipulation, and misdirection.

The concept of deception has been a subject of study and thought for centuries [125, Bk. III], [126]. Moreover, there exist numerous studies of deception and manipulation detection, yet deception recognition remains a difficult challenge. Several approaches have the potential to yield some success in automated identification, including recognizing linguistic features such as misleading language [127], [128], ambiguous syntax [129] and selective vocabulary [130]–[133]. Additionally, applying document graphs and argumentation structures [134]–[137] to articles may help identify deceptive arguments.

4.3.3. Misdirection

The validity of the text may be evaluated from a few angles. Most straightforwardly, flaws in logic progression may be presented. Errors in logic are commonly referred to as fallacies [124]. It should be noted that fallacious and therefore potentially misleading arguments are not evaluated for the author's intention—the fallacies could be due to weaknesses in the author's logic, or they could be due to intentional deception. Regarding the consumption of fake news, the intent is secondary to understanding whether valid arguments are being presented. This may be the rarest method for misleading readers, perhaps more common in scientific hoaxes than in political writings. For the current study, most of the focus will be on political writings, as that is where the greatest volume of fake news data currently resides.

4.3.4. Bias

In addition to that direct approach, however, others such as the objectivity and balance of the discussion may be explored. News articles often lack any real argumentation, and are more properly about reporting information, not debating issues. However, those reports can either provide information from a balanced perspective, reporting on multiple viewpoints, or they can be provided from a biased perspective. In [42], the authors define objectivity as comprised of truthfulness, neutrality, and detachment. They argue, however, that objectivity is an unobtainable ideal, one that has essentially been abandoned by journalism. In contrast, it might be argued that (objective) journalists have been abandoned by the consuming public [33], [34], [42, Ch. 3]. If slanted, self-congratulating news is in fact what the paying public demands, then that is a social ill that wants addressing, but again must remain outside the scope of this effort. Here, the goal is to aid concerned news consumers in recognizing how balanced (or unbalanced) is their news diet.

Because bias can be communicated both through argument structure and language choice, bias recognition could be derived from argument analysis and from language analysis. Both of these methods of presenting (and detecting) bias would be considered presentation bias, as defined in [138, p. 134]. The selection bias defined by Groeling is not addressed in this research, as this research endeavors to recognize fake news in individual news pieces, whereas selection bias addresses what news is being presented versus potential news that is not selected for publication. Nevertheless, selection bias clearly falls within the bias component of fake news, and could be detected in a similar manner, but just at a different scale and through the application of the fake news framework to a collection of documents individually and compiling statistics.

While a bias can be expected in what might be clearly labeled as an editorial in journalism, news should not be so slanted as to completely forego objectivity. Likewise, when consuming news stories, if an article abandons objectivity but does not represent itself as an editorial or opinion piece, the reader should certainly be aware, and most probably should be on guard. Consider the citation in Figure 4-2, for example. The argument diagram for it is revealed in Figure 4-5. While the article makes some attempt to report on both "sides" of the issue, citing Bryan Pagliano's lawyer's explanation for his client's no-show and the House Oversight Chairman's argument for requesting Mr. Pagliano to testify, the two sides are clearly not evenly discussed. From the diagram, one can see there are essentially eight propositions supporting the theme (the title of the article), versus only three to counter it.

Compare that with Figure 4-12 and Figure 4-6. There are 15 clearly supportive propositions for the thesis statement, and 7 refutations. While this is still perhaps not a balanced news article, it gives the appearance that it is more balanced than the one in Figure 4-2. It is interesting to note, however, that both real news articles clearly state their thesis in their headlines and maintain their position throughout the documents. It is also interesting to focus on one proposition: "*Cuomo, a Democrat who has endorsed Hillary Clinton for president this cycle*". This is, perhaps, a prime example where the reader's premises might define the actual function of the statement. A Trump supporter might view this as support for the refutation, noting that Cuomo is a political opponent, or at least not a supporter, of Donald Trump. Equally, though, a reader who does not support Trump might instead view this as a refutation of Donald Trump's position, considering it as evidence that Trump's position is biased.

The news story in Figure 4-4 has a problem not only with balance, but with source support. It essentially provides only one external reference to the news story, citing a prominent conservative governor, with no discussion from the liberal point of view. While this example is built on complete fabrication, and thus would be difficult to gauge without fact-checking the event, the construction of the article also reveals itself to be considerably unbalanced. Figure 4-7 vividly illustrates the dramatic contrast between this fake news article and the two real ones. Only a single refutation is provided compared to twelve supporting propositions, and that sole refutation is only a disclaimer that an opposing viewpoint was unavailable.

4.3.4.1. Support Vector Classification (SVC) Analysis

Support vector machines (SVMs) have been demonstrated to be effective in pattern recognition, regression estimation, and solving linear operator equations [139]. More recently, SVMs have been successfully applied to text categorization [140] and classification [141]. In [142], the authors apply support vector classification (SVC) to automatic detection of fake *negative* hotel reviews. This is a direct follow-on to the study of fake *positive* hotel reviews provided in [143]. By limiting the study to not only reviews, but hotel reviews, the subject matter and therefore much of the language remains consistent throughout the database—advantageous to the study, but not useful for addressing the more general challenging of classifying fake news articles. It should be noted that additional research efforts are reviewed in Section 2.3, including investigations applying SVC, deep learning neural networks, and Doc2Vec [64], [65]. These approaches were also investigated in the prior work but resulted in similar overall findings and have not been reported here. The overall conclusion is much room for improvement remains for automatic

fake news recognition.

Table 4-3 reflects the replication results compared to the originally published results. Interestingly, the results from a straightforward implementation using the SVC module in

*Table 4-3: Replication of Research Published Previously**

			TRUTHFUL			DECEPTIVE		
Model	Test Sentiment	Accuracy	P	R	F	P	R	F
Original* results, trained on positive and negative reviews	POSITIVE (800 reviews, Cross Val.)	88.4%	87.7	89.3	88.5	89.1	87.5	88.3
	NEGATIVE (800 reviews, Cross Val.)	86.0%	85.3	87.0	86.1	86.7	85.0	85.9
Replication	Both (1600 reviews, 0.8 train/test split)	90%	94	87	90	86	93	90
Replication with 5-fold cross-val	Both (1600 reviews, no test reserved)	69%	70	67	68	68	71	69
Replication with 10-fold cross-val	Both (1600 reviews, no test reserved)	80%	80	79	79	79	80	80

* M. Ott, C. Cardie, and J. T. Hancock, "Negative Deceptive Opinion Spam," in *Proceedings of NAACL-HLT*, 2013.

Abbreviations - P: Precision; R: Recall; F: F-score

the Python scikit learn library[15] produced improved results on the original, when using an 0.8/0.2 train and test split on both positive and negative reviews. This advantage disappeared, however, when progressing to cross-validation classification, and so was possibly due to a particularly fortunate draw on the training and testing data. Regardless, the results with 10-fold cross-validation are comparable to the original study, though slightly less favorable. The difference could be lessened with additional tuning but was judged unnecessary for demonstrating a comparable implementation.

Table 4-4 reveals that for training on PolitiFact and testing BuzzFeed data, the binary classification accuracy is merely 64%. This could perhaps be improved with parameter tuning, but as the datasets are equally balanced between real and fake news, the achieved accuracy is only slightly better than an expected accuracy of 50% for random guessing. Much greater accuracy is required to successfully recognize fake news.

[15] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Table 4-4: Bigram SVC Binary Classification of Fake News

		TRUTHFUL			DECEPTIVE		
Model	Accuracy	P	R	F	P	R	F
BuzzFeed 10-fold CV prediction	65%	68	58	63	63	73	68
PolitiFact 10-fold CV prediction	76%	79	70	74	73	82	77
BuzzFeed predict; PolitiFact training	64%	62	73	67	67	55	60
PolitiFact predict; BuzzFeed training	74%	75	72	74	73	76	75

Abbreviations - P: Precision; R: Recall; F: F-score

4.3.5. Manipulation

Another possible approach is to analyze news stories for their manipulative content, especially from the angle of emotional manipulation. Walton [144] proposed a method for determining when persuasive arguments can be classified as deceptive. As Walton states, "By a careful selection of terms with emotive meanings, a speaker can make an argument more powerfully persuasive by evoking positive or negative attitudes of the audience." Thus, through careful use of connotative language, an author can manipulate while not deviating from the denotative truth. This demonstrates why simple fact-checking is not enough for guarding against fake news. Both misleading arguments and manipulative

language can be employed to turn fact-based statements into fake news.

As noted in [42, pp. 78–79], [144]–[146], the language used to report a news item can add bias, either intentionally or unintentionally. Furthermore, the research in [62], [63], [147] shows some promise in applying manipulative linguistic cues to recognizing deceptive, misleading, and fake news through the use of machine learning or deep learning neural network models. A significant portion of the remaining research for the current study will be to explore the most effective way to incorporate such findings into automated processes for flagging manipulative text in news articles.

4.3.5.1. Linguistic Inquiry and Word Count (LIWC) Results

Linguistic Inquiry and Word Count (LIWC) is a method and commercial product for evaluating the psychometric property of texts basically through tracking the count of words used in the documents [148], [149]. Its goal is to provide an "efficient and effective method for studying the various emotional, cognitive, and structural components present in individuals' verbal and written speech samples…" [149]. It has been shown to be somewhat effective in distinguishing liars from truth-tellers [133], succeeding 61% of the time for the binary classification task. More importantly, the research yielded some insights into the psychometric differences between the two groups. Since this study was accomplished in 2003, the LIWC dictionary and software has been significantly updated, with new summary variables to explore. As previously discussed, the authors of [62] and [63] also made use of LIWC, resulting in some useful insights into misleading language. In order to take advantage of the full capability of this software, a license must be purchased

or rented. An abbreviated version is available online[16] for short texts of no more than 5000 characters, which was employed for this study to gauge whether further efforts using this tool might be gainful.

Table 4-5: LIWC Results for Sample Real and Fake Stories

Traditional LIWC dimension	Real data	More Real Data	Fake data	Avg for Social media	Avg for professional or scientific writing
I-words (I, Me, My)	0.9	0.7	0.0	5.51	0.63
Social Words	11.2	12.3	5.0	9.71	7.62
Positive Emotions	0.9	2.0	0.5	4.57	2.32
Negative Emotions	3.7	3.1	2.0	2.10	1.45
Cognitive Processes	8.4	6.5	9.5	10.77	7.52
LIWC Summary Variables					
Analytic	98.4	50.1	91.3	55.92	92.57
Clout	83.7	92.4	56.0	55.45	68.17
Authenticity	1.6	10.0	25.2	55.66	24.84
Emotional tone	2.0	12.3	8.1	63.35	43.61

[16] http://liwc.wpengine.com/

Table 4-5 shows the results of running the online tool on the three sample texts in Figure 4-2, Figure 4-3, and Figure 4-4. The LIWC dimensions of *Social Words* and *Positive Emotions* show some possibilities, as do the summary variables *Analytic* and *Clout*. This is of course strictly anecdotal at this juncture, as only three sample texts were tested at this time, but these preliminary results appear to warrant further investigation with the full software license, which will also provide a much larger selection of dimensions and summary variables than the online version.

4.4. Fake News Model Conclusions

While much promise is revealed in the proposition and investigation of the fake news framework, it has also revealed some serious hindrances. Firstly, argument mining remains particularly intractable [150], with the many advances being limited to domain-specific arguments, notably with respect to legal arguments [151]–[156]. As large language models (LLMs) take hold, argument mining will hopefully advance rapidly into a tractable problem. While the approach of examining the validity and solidity of arguments presented within (fake) news articles remains a substantial goal, for the present, even with the aid of LLMs, it remains effectively out of reach.

In contrast, while the power of LLMs can aid mightily in the effort to categorize and classify news as real or fake (or somewhere between), that same tool can be employed as a weapon to constantly invent and adapt new approaches to producing fake news to defeat efforts at detection. The nature of this ongoing battle and its many intricacies is outlined effectively in [118]. This has all the hallmarks of a long-running arms race, where each new effort to identify and dismiss either perpetrators of fake news or the fake news content itself will be quickly met with adaptations to further obscure the deceitful nature of the

producers or the product. Pitting deep learning (or other advanced) AI fake news detectors against generative AI fake news generators promises to be a perpetual battle for some time, with money and resources largely dictating the outcomes of each battle. While it might be a necessary endeavor to engage in that struggle, it outpaces the capacity and usefulness of this researcher to contribute meaningfully to that battle, at least in the short term. Completion of the proposed fake news framework could very well assist in this effort, but unfortunately the resources required to keep it relevant are not currently available. Moreover, as revealed in recent research [157], steps taken to prevent or prohibit access to misinformation can instead backfire. Further labeling news as fake or real may only fuel the fire of those with opposing views.

A more practical and manageable approach is, instead, to identify the hallmarks of a fake news breeding ground—the regions of the internet where, once fake news "seeds" are planted, the contagion of misinformation grows and thrives. What we now propose as an immediate necessity, more dire and essential than identifying the fake news itself, is identifying where and when we can expect fake news to gain a foothold, and thus try to combat it early, or, at the very least, detect and monitor it as it grows. Thus, we now propose an approach to identifying echo chambers, where fake news might be introduced and readily repeated. The next section will introduce a methodology for doing exactly that. Note though, that it does not remove the need for the fake news framework as a means to classify fake news—that in time will be essential as well. However, we suggest that even more immediate is the need to understand and identify the conditions under which a social media platform, or special interest group on the same, might offer ideal conditions for the dissemination and replication of fake news.

5. Echo Chambers and Gravity Wells

If we accept the assertion that fake news is often incubated in social structures commonly known as echo chambers, then successfully identifying echo chambers would be a great stride towards combating the ill-effects of fake news. Certainly, research has shown that the echo chamber effect is often reinforced by social media algorithms [29], [68], [158], [159].

As identified in the **Introduction**, we propose that a social media group may be effectively modeled as a gravity well [30], where an echo chamber might reveal itself by its overwhelming ability to attract and retain users within such a well. Furthermore, we postulate that the composition of that gravity well would be self-sorting, where members within the echo chamber would necessarily maintain similar views to remain comfortable within the well and not be drawn to some other nexus. An echo chamber in a social media interest group essentially *captures* its audience, where the attractive force of the well is composed of elements of the social media platform, the precise topic of the forum, and characteristics of the audience characteristics of the audience. It is this insight into what creates the overwhelming force of echo chambers, as well as the simulation of social media interest groups as gravity wells, that constitutes a significant contribution to the study of fake news in general and of echo chambers more specifically.

In this section, we will outline the functioning of the gravity well model and its employment for the purposes of this experiment. To accomplish this, we will discuss the formation of the social media interest group gravity well and then the subsequent tuning of variables designed to identify the contribution of different aspects of the modeled social media groups. It will be useful to first introduce the data used for the entirety of this

experiment.

5.1. Model Data

After considering numerous social media platforms for our initial experiment, including Facebook, YouTube, Twitter, and Reddit, we chose to use Reddit data due to the (formerly) publicly available archives of subreddit submissions[17] and comments[18] online via the Pushshift archives. Certainly, other social media providers have publicly accessible application programmer interfaces (APIs) and some also have data repositories, which will prove useful for future research. Though some researchers [29], [73], [158] have found evidence that Reddit may be less likely than other social media platforms to encourage the formation of echo chambers, it is our expectation that the phenomenon nevertheless exists on Reddit, though less so than on other platforms.

For this effort, we accessed the data repositories to obtain all submissions and comments for all users in a target set of subreddits for an entire month. The selection of these subreddits was initially aimed at some of the most popular and active subreddits, e.g. 'memes' and 'Market76', but it was determined through experimentation that the volume of data (607/6639 and 1134/2839 posts/comments per day for 'memes' and 'Market76', respectively) would require specialized handling—a focus reserved for later efforts. Moreover, simply selecting from the most popular subreddits yielded topics that were popular, but not necessarily rich in conversation and opinions, as evidenced by the

[17] https://files.pushshift.io/reddit/submissions/

[18] https://files.pushshift.io/reddit/comments/

aforementioned ‘memes’ and ‘Market76’. The subreddit ‘memes’ includes a rule that specifies “No memes that are text only.” This of course limits the amount of conversation content that can be extracted from that subreddit’s posts. As another example, the subreddit ‘Market76’ is “A subreddit dedicated to trading for Fallout 76,” a popular computer game. Thus, it is a subreddit which focuses on material trading, not conversation.

Instead, we targeted average-sized but clearly active subreddits for this initial effort, such as ‘science’, with a manageable 42/923 posts/comments per day. Apart from one subreddit, namely ‘globeskepticism’, all subreddit data came from the same time span, for the month of January in 2019. Because ‘globeskepticism’ is a somewhat new subreddit, the data for it in 2019 lacked volume. Consequently, we used data from June of 2020 for the ‘globeskepticism’ subreddit. In total, we ran our simulation for 19 different subreddits, selected for their practicable size and range of topics, with the deliberate intention of including both political and apolitical interests.

The subreddits are listed in Table 5-1. Note that there are many other data items available. beyond just the text of posts and comments particular to Reddit, such as submission scores, but the goal here is to remain as social media platform agnostic as possible, so that a similar treatment may be applied to other social media platforms in future. The Python 3.8 code used to retrieve and preprocess the subreddit data can be found in Appendix B-1 and Appendix B-2.

Table 5-1: Average TSM for modeled subreddits

Subreddit	mass	coefficient of variation	std dev (TM = 6)	Avg TSM (TM = 6)
SandersForPresident	5558	0.068623	4.386	63.91
flatearth	1569	0.026245	2.901	110.52
trump	410	0.199327	49.252	247.09
globeskepticism	211	0.230475	92.252	400.27
science	34473	0.562384	200.495	356.51
cars	21296	0.338548	270.981	800.42
Republican	2170	0.900966	1492.658	1656.73
SocialDemocracy	157	0.899004	1652.935	1838.63
Freethought	155	0.674719	970.306	1438.09
travel	12175	0.314357	568.333	1807.92
math	5951	0.304073	802.086	2637.81
NeutralPolitics	1822	0.842791	1997.171	2369.71
PoliticalDiscussion	5709	0.393189	982.197	2498.03
democrats	3279	0.332461	1244.097	3742.08
hiking	3309	0.258738	994.055	3841.94
republicans	84	0.864630	2821.868	3263.67
mlb	1641	0.392950	2038.180	5186.87
progressive	639	0.591066	3370.690	5702.73
AmericanPolitics	141	0.318009	1837.753	5778.94

5.2. Gravity Well Model

In examining the characteristics of echo chambers, it was determined that identifying the requirements for the formation of an echo chamber would be necessary to facilitate modeling social media groups to ascertain the presence, or absence, of an echo chamber. Following on our agreed definition of an echo chamber [28, p. 76], the keys of "magnifying" and "insulating" appeared central to these requirements. Considering those key terms, we posit that the foundation for the echo chamber effect is formed by three pillars that when combined create a synergistic effect difficult to resist for many social media users. These three pillars are: 1) technology, which acts as magnifier and insulator; 2) topic, which serves as the initial draw and holding force; and 3) confirmation bias, which reflects the individual participant's "seeking or interpreting of evidence in ways that are partial to existing beliefs, expectations, or a hypothesis in hand" [160].

As with the genesis of physics exploration, we begin our gravity well model using a Newtonian approach, with the aim of adding complexity as need arises. To start, we employed Newton's law of universal gravitation [161, p. 141]:

$$F = G\frac{m_1 m_2}{r^2} \qquad (1)$$

where G is the universal gravitational constant, m_1 and m_2 are the masses of the bodies involved, and r is the distance between the bodies. Translating this to the context of social media interest groups, we conceive that m_1 would be the mass of the gravity well m_{w}, nominally represented by the number of social media users (hereafter referred to as "simulation agents" or simply "agents", being constructs of individual user data retrieved from social media) subscribed to or active in the interest group, while m_2 would be

represented by the mass, m_a, of a single agent of interest, to which we correspondingly affix a unitary value. Regarding the distance separating the masses, we conceived that the distance would be well represented by determining the *distance* between an individual and the aggregate echo chamber with respect to opinion or sentiment.

With respect to this measure, by comparing the sentiment of an individual agent's posts to the average of other agents' posts within the same subreddit, we expected to arrive at a reasonable estimation of the *social* or *opinion* distance r between the overall subreddit and the target agent, i.e., between m_1 and m_2 in (1). Ultimately, we employed the ratio of the *disagreement* between posts to the *affinity* of the agent for the overall topic of the social media interest group, i.e., for the subreddit topic. We will now explain how we obtain the disagreement and affinity measures.

To accomplish these measurements, we employed the language representation model called Bidirectional Encoder Representations from Transformers (BERT). BERT is a pre-trained transformer-based natural language processing tool created by authors at Google. Unlike many pre-trained language models, BERT uses bidirectional encoding to capture the context of usage of a given word, thus enabling it to distinguish between alternate uses of identical words. For our purposes, obtaining text embeddings for sentences provided us the capability to compare different posts, which we accomplished using a Sentence-BERT [162] library provided for Python.

The disagreement measure was computed by first obtaining BERT text embeddings for all posts made within the time span of the initial data for the simulation (1 month). Each text embedding was then pairwise compared with every other text embedding using cosine

similarity. The disagreement for a single post was derived by taking the average cosine similarity of that post with all posts made by the entirety of the social media group, in this case various subreddits, and then taking its additive inverse, as the range of the similarity varies from -1 to 1. All such values are then averaged for all posts by a single author to obtain that author's disagreement score.

Likewise, we used BERT to estimate the affinity of an individual user to the topic of a subreddit by averaging the similarity between all the user's posts and the published subreddit description. It is recognized that this measure of affinity has some obvious shortcomings with respect to any posts made that are essentially nonreflective of the subreddit description, not to mention that some subreddit descriptions are not particularly reflective of the group topic, but it is expected this shortcoming would be shared by all users. This has been flagged as an issue for further refinement in future efforts. With this approach, we arrive at a new "radius" based on the ratio of disagreement to affinity that we label r_{da}.

The final remaining component within (1) is the universal gravitational constant G. As with the physical proportionality constant, it must be obtained through observation. For the purposes of our simulation, it serves as a tuning point for our simulation equation to return manageable results for the magnitudes and ranges of our input data. Thus far, it has been tuned to a value of $G_w = 10^{-6}$, though it may very well need adjustment as data from different social media platforms are added to the mix.

Taking the analogies discussed thus far, we arrive at a new equation for the force in the gravity well simulation:

$$F_w = G_w \frac{m_w m_a}{r_{da}^2}, \qquad (2)$$

where $G_{\mathrm{w}} = 10^{-6}$, m_{w} is the number of agents in the well, $m_{\mathrm{a}} = 1$, and r_{da} is the ratio of the disagreement of the agent's posts with the entirety of posts in the well to the affinity of the agent's posts to the published subject of the social media group.

Returning now to the implementation of our three pillars of an echo chamber, a user's confirmation bias is intended as a modifier affecting the calculation of their affinity and thus affecting the value of r_{da} in (2). Confirmation bias's effect is currently earmarked for future study, and as such has been fixed at unity for all agents for the duration of this experiment. It was necessary to postpone implementation of confirmation bias due to the required effort to introduce a measure of a tendency towards confirmation bias that was both effective and translatable across most, if not all, social media platforms. Our survey of materials related to confirmation bias and social media revealed a paucity of results, where more studies like that reported in [163] would be of most benefit, by relating individual characteristics to a preference for confirmation bias. To embark on such studies ourselves would be an unwelcome distraction from the main thrust of this thesis, and we hypothesized that the model could provide meaningful insights without that pillar implemented, though it might necessarily limit the accuracy of prediction of individual behaviors.

To incorporate the effects of the other two pillars, technology and topic, a technology modifier (TM) and a topic source modifier (TSM) were introduced into (2), such that their magnitudes would be *inversely* proportional to the effect of that pillar on the echo chamber. TM, being representative of the magnifying and insulating action of the echo chamber,

logically affects the mass of the echo chamber, m_1. Likewise, TSM amplifies the affinity a user has for the social media group topic and thus affects the value of r_{da}. We considered implementing these modifiers instead as directly proportional to their effects by inverting them and limiting their ranges to $0 \leq TM \leq 1$ and $0 \leq TSM \leq 1$, which would have been extremely convenient for tuning, but chose to avoid the potential for complications with analysis and manipulation should some of these values approach zero. Incorporating these adjustments into (2) yields:

$$F_w = G_w \frac{TM \cdot m_w \cdot m_a}{\left(\frac{r_{da}}{TSM}\right)^2}, \qquad (3)$$

Figure 5-1 provides a visual overview of the gravity well construction.

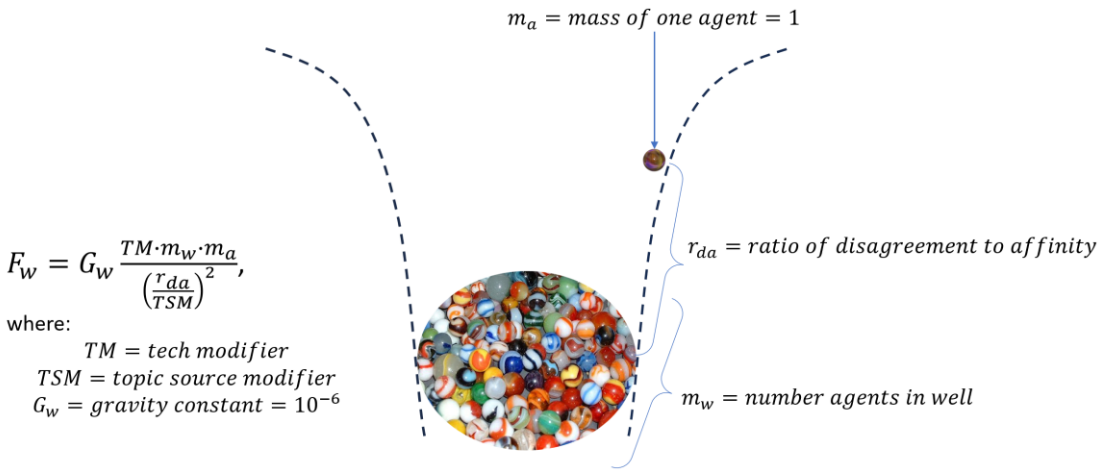

Figure 5-1: Gravity well structure

All the above outlines the basic construction of the gravity well simulation, except for what the equation yields. In the simulation, the expectation is that all agents within the gravity well at initialization, i.e., all users active in the modeled subreddit, will remain within the well until such time as the value F_w in (2), that is the simulated force of gravity, falls below a threshold and permits the agent to escape the well. Here again, it was

necessary to choose a threshold which worked well with the magnitude of values encountered during the simulation. This value was held constant for the duration of this experiment but may be adjusted in future as additional social media platforms are modeled. The Python code for the gravity well simulation can be found in Appendix B-3.

5.3. Tuning the Model

Having established the construction of the social media interest group gravity well, we can now turn to the use of the well to determine values of interest for the selected social media platform, Reddit. For each subreddit, a month's worth of submissions and comments from all users in the subreddit are used to initialize agents for that simulated gravity well. The simulation is then run for 90 iterations, where, for each iteration, a subset of the agents is selected to attempt to exit the echo chamber. Currently, the only deliberate updates to an agent's force equation (1) during simulation iterations are updates to an agent's affinity and disagreement. An agent's disagreement randomly varies around its original value. Individual affinity is likewise allowed to randomly vary, but that variation is designed to favor increasing over time, as research has indicated that biases tend to become more extreme within like-minded groups [164]–[167]. Additionally, though, the mass of the entire gravity well, m_1 in (1), will naturally reduce as agents leave the echo chamber—for the current research, agents are allowed to leave but do not consider rejoining, though that capability exists and will be explored in future efforts.

When tuning, we ran multiple instances of each social media group, i.e., for each subreddit. In tuning, we focus on the turning point where the first agents go from remaining in the gravity well to exiting. This is necessary, as either outcome—all agents remain in the well versus all agents exit the well—can be associated with an infinity of variable

values. The goals in tuning via running multiple simulations are, in turn:

1. Determine a value for the universal gravitational constant G that produces *reasonable* results for the range of data being explored and fixed values of TM and TSM.

2. Simultaneously tune all subreddits, using a fixed *TSM* value, to obtain a common value for *TM*, which is intended to represent the influence Reddit's technology has on the echo chamber effect. To accomplish this, we set a goal of finding the minimum TM value that would result in approximately 10% of the agents escaping the gravity well within the 90 iterations. We were essentially seeking the turning point of each subreddit's well going from holding all agents to initially allowing a few to escape, then selecting the minimum among these TM values to represent the overall technological effect of Reddit, with any differences between subreddits attributed to each subreddit's attraction to its members.

3. Finally, tune for a unique TSM value for each subreddit. To that end, once a TM value was decided, we initiated 100 runs of the model for each subreddit, again for 90 iterations using that established TM value. The goal now was to tune the TSM for each run to allow approximately 1% of the agents to escape and thus again find the turning point. It is hypothesized that identifying the average TSM value for each subreddit's turning point will yield insights into the nature of the gravity well associated with each subreddit.

Figure 5-2 depicts the tuning process for the model, while the Python 3.8 code used

for tuning the simulation can be found in Appendix B-4. In the next section, we will discuss the results of running our gravity well simulation of echo chambers on subreddits.

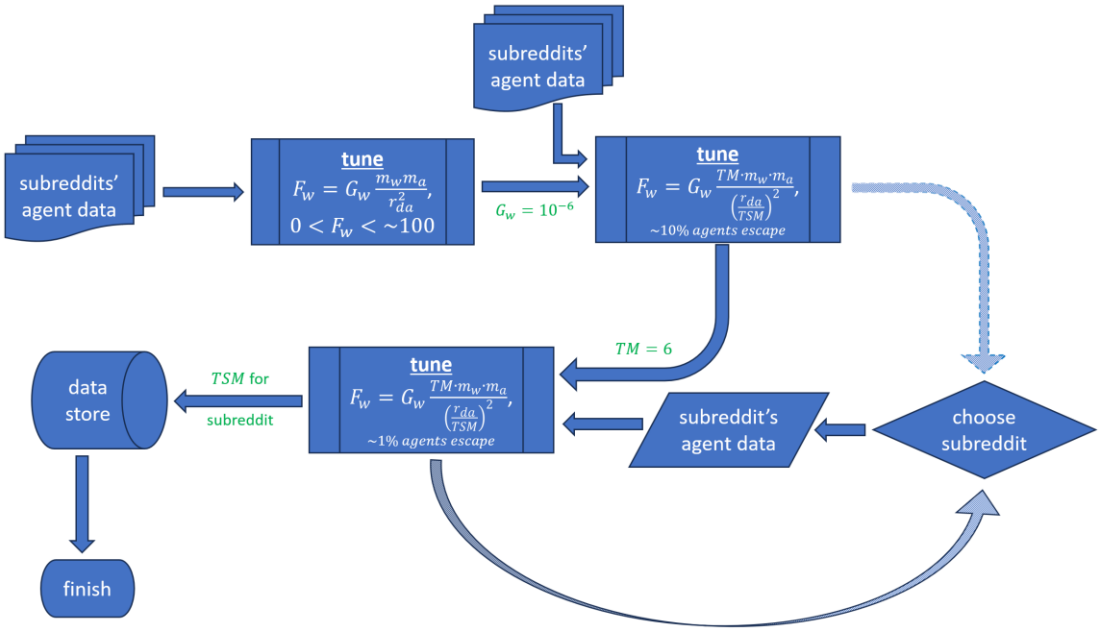

Figure 5-2: Tuning process

6. Gravity Well Simulation, Results, and Analysis

Having produced a gravity well simulation designed to model the users in social media interest groups, it yet remains to demonstrate this model and determine its efficacy in identifying those groups which have the hallmarks of an echo chamber or have the potential to become destructive echo chamber should fake news be introduced into those chambers.

6.1. Simulation

To begin tuning, we chose to tune the *TM* first, as we require a constant value for Reddit in general to obtain relative *TSM* values for each subreddit. Thus, we first fixed the *TSM* to a constant value of 25 to ensure we would obtain a *TM* for Reddit with sufficient magnitude to allow tuning for individual *TSM* scores for each subreddit. It is the relative measures for *TM* and *TSM* that are of interest, so fixing the *TSM* at 25 was simply a convenient nominal value which we knew would be replaced with tuning later. Next, we tuned across all 19 subreddits, selecting the minimum *TM* value thus obtained as the *TM* for Reddit in general. After 100 independent tuning runs for all subreddits using $G_w = 10^{-6}$ and $TSM = 25$, we arrived at a minimum value of 6 for the technology modifier (TM) for Reddit. As this value has no units and is simply intended to serve as a relative measure of the impact this social medial platform is having on the echo chamber, this value will suffice for our current experiment. In future, as more subreddits or other social media platforms are introduced into the model, this value may require adjustment. Using these values, $G = 10^{-6}$ and $TM = 6$, we then ran 100 simulations of each subreddit, to determine the average TSM value that resulted from tuning according to 5.3.

6.2. Results

The results from tuning TSM values for all the subject subreddits can be viewed in Table

5-1. As is evident in the table, with the subreddits sorted by ascending average TSM and thus decreasing gravitational affect, the TSM does not linearly correspond to the mass (or effectively the size) of the subreddit. The coefficients of variation and standard deviation are likewise relatively evenly distributed through the sorted *TSM* values, with no obvious clustering related to the TSM results. There are clearly other things at play, with average TSM values ranging between 63.91 and 5778.94 for these 19 subreddits. The issue we wish to explore is whether these distinctions illuminate the occurrence of echo chambers in these subreddits. Our expectation is that subreddits with smaller TSM values would suggest a more powerful hold on members, keeping them bound to the gravity well. Thus, those subreddits would be more prone to forming echo chambers. In contrast, the subreddits with larger TSM values would essentially allow members to come and go at will. As mentioned previously, this does not imply the potential echo chamber is harmful or unhealthy, just that it has the hallmarks of an echo chamber.

6.3. Analysis

6.3.1. Validity of Simulation

While the simulation seems to perform reasonably and in an expected manner, validation of the gravity well's performance with respect to real data is needed to prove our primary hypothesis.

Hypothesis 1: An interest group on a social media platform can be effectively modeled as a gravity well.

To this end, we compared the exiting behavior of each agent in the simulation to the exiting behavior of its associated subreddit user. Table 6-1 highlights the results of this

Table 6-1: Mean Absolute Percent Error of Agent Exit Ordering

subreddit	MAPE (%)	TSM	Mass
Freethought	1.15	1438	155
republicans	1.55	3264	84
AmericanPolitics	1.62	5779	141
globeskepticism	2.40	400	211
SocialDemocracy	2.49	1839	157
trump	2.95	247	410
travel	3.02	1808	12175
Republican	3.10	1657	2170
NeutralPolitics	3.13	2370	1822
math	3.15	2638	5951
progressive	3.17	5703	639
flatearth	3.20	111	1569
mlb	3.26	5187	1641
democrats	3.93	3742	3279
PoliticalDiscussion	4.09	2498	5709
hiking	4.16	3842	3309
cars	4.35	800	21296
SandersForPresident	4.37	64	5558
science	4.90	357	34473

comparison, where we can see that the mean average percent error (MAPE) of exiting behavior for all agents in each subreddit falls well below 5%, thus convincingly demonstrating that the gravity well simulation of the subreddits performs realistically and emphatically confirming *Hypothesis 1*. The Python 3.8 code used for calculating the MAPE values in Table 6-1 can be found in Appendix B-5 and Appendix B-6.

6.3.2. *Statistical Significance of TSM Values*

We first wish to establish if there is a statistically significant variation in calculated *TSM* values for the 19 subreddits. This is in essence our second hypothesis.

> *Hypothesis 2: The calculated TSM values serve as statistically significant discriminators for each*

We can establish this through an analysis of variance (ANOVA) calculation. With a null hypothesis that all the subreddit mean *TSM* values are equal and thus not statistically significant, we obtained a $pvalue = 0.0$. Thus, we reject the null hypothesis and conclude

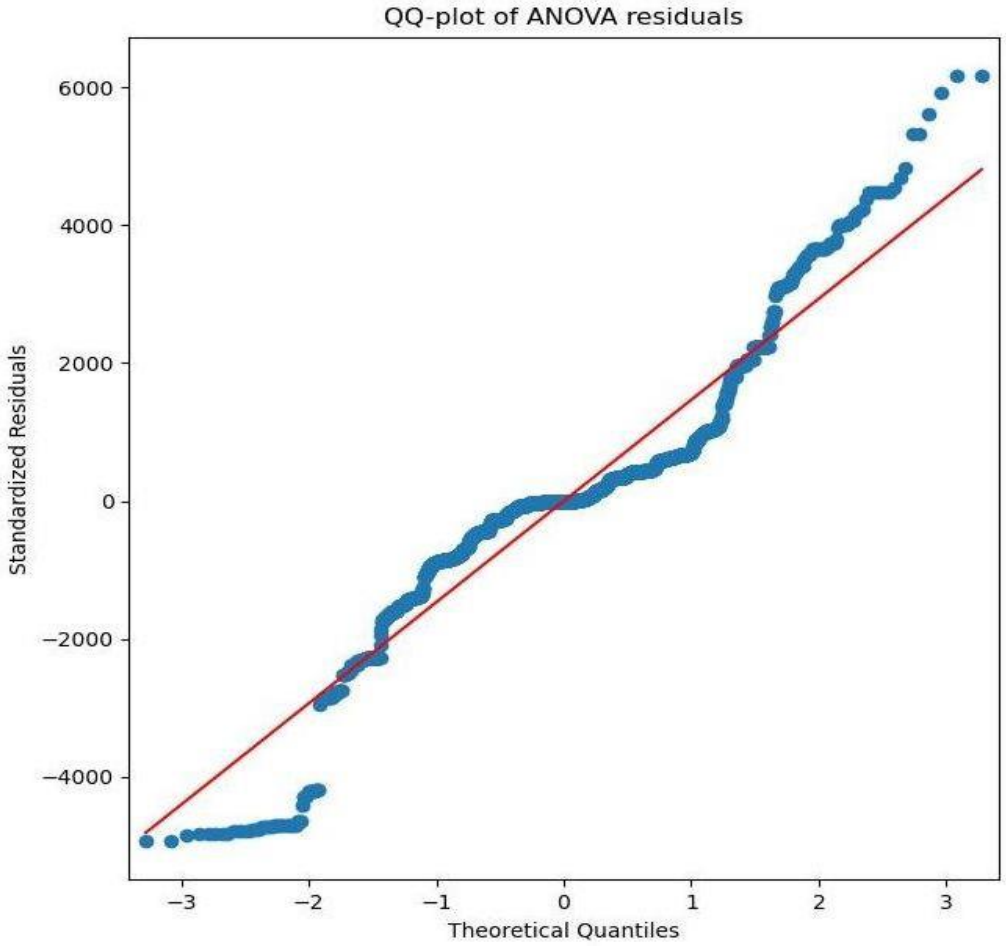

Figure 6-1: QQ Plot of ANOVA residuals versus normal line

there exists a statistically significant variation in the *TSM* values among the 19 subreddits. This result therefore confirms *Hypothesis 2*.

To confirm that the data reasonably conforms to the ANOVA assumptions of normality, we first performed a Shapiro-Wilk test for normality by applying the scipy.stats.shapiro() function to the ANOVA residuals. This yielded a disturbingly small p-value of 1.059e-33, indicating that the null hypothesis of normality should be rejected. However, we also generated a QQ-plot (Figure 6-1) and a histogram (Figure 6-2) of the residuals from the ANOVA analysis. The residuals in Figure 6-1 follow the reference normal line in the QQ-plot; likewise, the histogram of residuals in Figure 6-2 looks approximately normal. Considering that research has shown that ANOVA is reasonably robust to nonnormality [168] and unequal variances [169], we are confident that ANOVA is appropriate for this analysis.

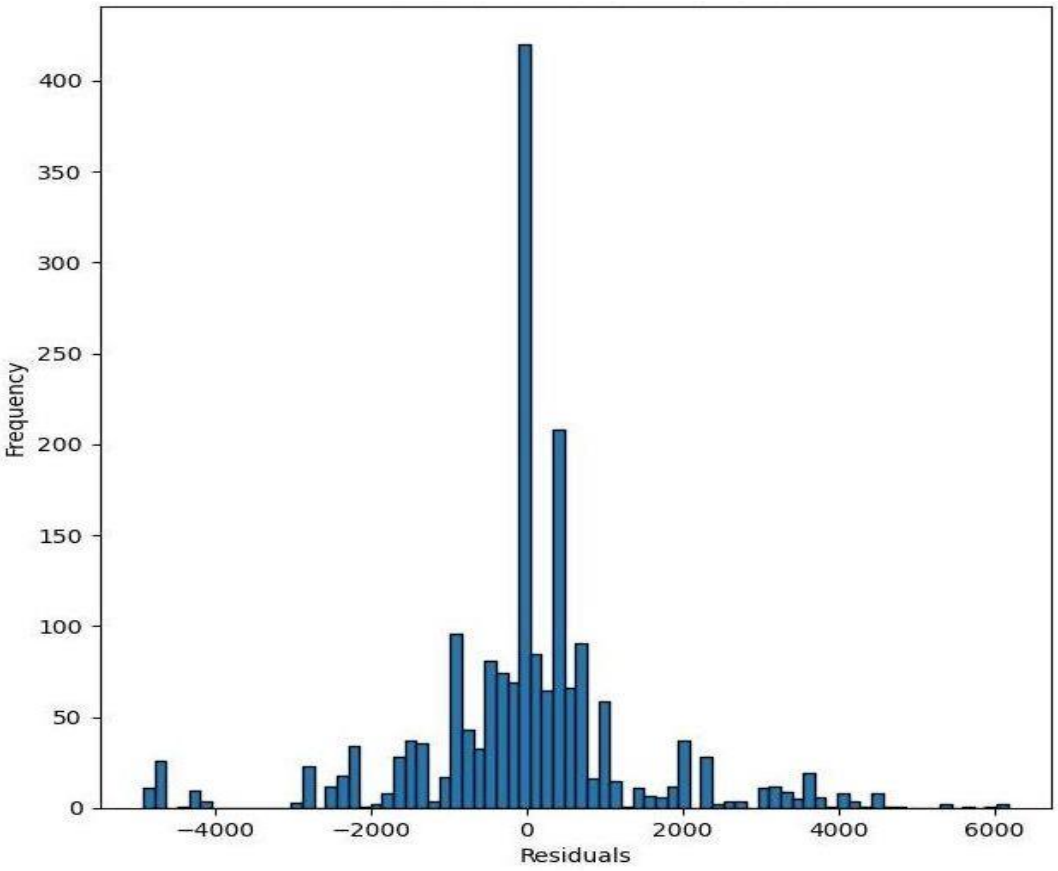

Figure 6-2: Histogram of ANOVA Residuals

Table 6-2: Multiple comparison of means for subject subreddits

TSM-1	group1	TSM-2	group2	meandiff	p-adj	lower	upper	reject
64	SandersForPresident	111	flatearth	46.61	1	-688.0985	781.3185	FALSE
64	SandersForPresident	400	globeskepticism	336.36	0.9848	-398.3485	1071.0685	FALSE
64	SandersForPresident	357	science	292.6	0.9969	-442.1085	1027.3085	FALSE
64	SandersForPresident	247	trump	183.18	1	-551.5285	917.8885	FALSE
111	flatearth	400	globeskepticism	289.75	0.9973	-444.9585	1024.4585	FALSE
111	flatearth	357	science	245.99	0.9997	-488.7185	980.6985	FALSE
111	flatearth	247	trump	136.57	1	-598.1385	871.2785	FALSE
357	science	247	trump	-109.42	1	-844.1285	625.2885	FALSE
400	globeskepticism	357	science	-43.76	1	-778.4685	690.9485	FALSE
400	globeskepticism	247	trump	-153.18	1	-887.8885	581.5285	FALSE
800	cars	111	flatearth	-689.9	0.0965	-1424.6085	44.8085	FALSE
800	cars	400	globeskepticism	-400.15	0.9196	-1134.8585	334.5585	FALSE
800	cars	357	science	-443.91	0.8221	-1178.6185	290.7985	FALSE
800	cars	247	trump	-553.33	0.4402	-1288.0385	181.3785	FALSE
1438	Freethought	1657	Republican	218.64	0.9999	-516.0685	953.3485	FALSE
1438	Freethought	1839	SocialDemocracy	400.54	0.9189	-334.1685	1135.2485	FALSE
1438	Freethought	800	cars	-637.67	0.1889	-1372.3785	97.0385	FALSE
1438	Freethought	1808	travel	369.83	0.9603	-364.8785	1104.5385	FALSE
1657	Republican	1839	SocialDemocracy	181.9	1	-552.8085	916.6085	FALSE
1657	Republican	1808	travel	151.19	1	-583.5185	885.8985	FALSE
1839	SocialDemocracy	1808	travel	-30.71	1	-765.4185	703.9985	FALSE
2370	NeutralPolitics	2498	PoliticalDiscussion	128.32	1	-606.3885	863.0285	FALSE
2370	NeutralPolitics	1657	Republican	-712.98	0.0694	-1447.6885	21.7285	FALSE
2370	NeutralPolitics	1839	SocialDemocracy	-531.08	0.522	-1265.7885	203.6285	FALSE
2370	NeutralPolitics	2638	math	268.1	0.999	-466.6085	1002.8085	FALSE
2370	NeutralPolitics	1808	travel	-561.79	0.4101	-1296.4985	172.9185	FALSE
2498	PoliticalDiscussion	1839	SocialDemocracy	-659.4	0.1447	-1394.1085	75.3085	FALSE
2498	PoliticalDiscussion	2638	math	139.78	1	-594.9285	874.4885	FALSE
2498	PoliticalDiscussion	1808	travel	-690.11	0.0962	-1424.8185	44.5985	FALSE
2638	math	3264	republicans	625.86	0.2166	-108.8485	1360.5685	FALSE
3742	democrats	3842	hiking	99.86	1	-634.8485	834.5685	FALSE
3742	democrats	3264	republicans	-478.41	0.7147	-1213.1185	256.2985	FALSE
3842	hiking	3264	republicans	-578.27	0.354	-1312.9785	156.4385	FALSE
5187	mlb	5703	progressive	515.86	0.579	-218.8485	1250.5685	FALSE
5779	AmericanPolitics	5187	mlb	-592.07	0.3101	-1326.7785	142.6385	FALSE
5779	AmericanPolitics	5703	progressive	-76.21	1	-810.9185	658.4985	FALSE

6.3.3. Differentiation of TSM Values

With this outcome, we are assured there are significant differences in the subreddits with respect to their TSM values, but we do not know *which* subreddits differ from one another. To learn this, we performed a multiple pairwise comparison of means using Tukey's HSD test. Of the 171 pairings, a full 135 rejected the null hypothesis with $p = 0.05$, indicating they were significantly different. The remaining 36 pairs appear equivalent and provide the most interesting insights into the simulation outcome. The results for the 36 equivalent pairs that failed to reject the null hypothesis are provided in Table 6-2 – we omit the results for the 171 mismatched pairs due to space considerations.

The results in Table 6-2 are sorted in ascending order of TSM for the first subreddit of each pairing. Further, the magnitudes of the TSMs are highlighted on a color spectrum beginning with green for minimal values, yellow for mean values, and red for maximal values. This contrast aids observing the trends in the results. We have two extreme groupings, those pairs with minimal TSM values in the first 14 pairings and those with maximal TSM values in the last 6 pairings, with the remaining 16 moderate pairs standing between. Once again, we note that a small value for TSM indicates a disproportionate ability to attract and keep agents within the well. Thus, we conclude that the first 14 pairs have the greatest potential to form echo chambers.

The question now becomes, are these results reasonable and informative? While we do not currently have quantifiable measurements to inarguably determine if a particular subreddit is functioning as an echo chamber, we can make some observations on the makeup of these groupings.

The first 14 pairings involve six unique subreddits, which we can think of as a minimal *TSM* group. To understand the import of this group, it is informative to examine the descriptions of the involved subreddits (Table 6-3). Reviewing the text, we might

Table 6-3: Minimal TSM subreddit descriptions

subreddit	description
SandersForPresident	Bernie Sanders 2024
cars	/r/Cars is the largest automotive enthusiast community on the Internet. We are Reddit's central hub for vehicle-related discussion including industry news, reviews, projects, videos, DIY guides, stories, and more.
flatearth	Is the Earth actually flat? Where's the edge? How come we don't fall out? What about gravity? Learn all of this and more at this very serious subreddit.
globeskepticism	This is a safe community to discuss the shape of the earth, skeptics and researchers welcome. We've examined all the evidence supporting spinning ball earth over the years and we're not convinced. Disrespectful contributors will be banned. Other conspiracy talk also welcome.
science	This community is a place to share and discuss new scientific research. Read about the latest advances in astronomy, biology, medicine, physics, social science, and more. Find and submit new publications and popular science coverage of current research.
trump	This community is for discussing the 45th US president and all things associated.

conclude that these subreddits make little to no effort to hide their bias or intent. It becomes quickly evident that the *cars* and *science* subreddits are for enthusiasts and, as such, are intended to hold well-established positions on their subject matter. They are thus unlikely to be echo chambers of concern, though still possibly meeting the most innocuous definition of an echo chamber. We can also likely add *flatearth* to the list of innocuous subreddits, as it is apparent from Table 6-3 that it is a satirical interest group regarding the implausibility of the earth being flat.

In contrast, though, the remaining subreddits in this group do have potential for not only being echo chambers, but also for being the type of echo chamber that could be abused

to introduce and incubate fake news items. This does not mean they have, just that they are primed for that possibility. Further analysis on the quality and reliability of the actual content being generated would be required to ascertain if these subreddits have been exploited for those purposes. These remaining three, *SandersForPresident*, *globeskepticism*, and *trump*, all have the potential to push a very targeted agenda.

Table 6-4: Maximal TSM subreddit descriptions

subreddit	description
democrats	The Democratic Party is building a better future for everyone and you can help. Join us today and help elect more Democrats nationwide! This sub offers daily news updates, policy analysis, links, and opportunities to participate in the political process. We are here to get Democrats elected up and down the ballot.
hiking	The hikers' subreddit.
mlb	Subreddit for Major League Baseball. From discussions, news, and highlights from all thirty MLB teams.
AmericanPolitics	A place to discuss the American political process, American political topics, the political parties, elected officials, candidates, and American foreign policy.
republicans	PRO-REPUBLICAN SUBREDDIT FOR ADULTS!
progressive	A community to share stories related to the growing Modern Political and Social Progressive Movement. The Modern Progressive Movement advocates change and reform through directed governmental action. The Modern Progressive Movement stands in opposition of conservative or reactionary ideologies.

Continuing to the maximal TSM group in the last six pairings, it also consists of six unique subreddits, despite being a smaller group of pairings. Those six subreddits appear in Table 6-4. It is interesting that this much smaller grouping has an equal number of unique subreddits as the minimal TSM group. While there is a preponderance of political subreddits in this second group, those political subreddits (*AmericanPolitics*, *progressive*, *republicans*, and *democrats*) are either more party-centered or intentionally party-agnostic, while the political subreddits in the minimal TSM group were more personality focused. The remaining two subreddits in this group are purely recreational. It is interesting that

these two recreational subreddits, *mlb* and *hiking*, should produce large *TSM* values, while the two recreational subreddits in the minimal *TSM* group, *cars* and *science*, produce the opposite.

To establish the likenesses and differences between the minimal and maximal *TSM* groups, we submitted the entirety of the posts from each subreddit, both original submissions and comments, to Google's Perspective Comment Analyzer (PCA) API[19]. As noted by the API website, the aim for the PCA is to use the power of machine learning to reduce online toxicity. In short, the PCA API seeks to rate text according to seven measures: toxicity, severe toxicity, insult, profanity, identify attack, threat, and sexually explicit. For our purposes, we focused on using the API to determine if one group tended to be more severe than another according to four of those labels: toxic, severely toxic, insulting, and threatening. Our expectation is that potentially harmful echo chambers would score higher on these indices.

The results of this analysis may be viewed in Table 6-5. Again, the subreddits are ordered by increasing *TSM* value to facilitate comparing the minimal and maximal *TSM* groups. It is immediately apparent that those groups do not separate themselves based on these negative criteria of being toxic, severely toxic, insulting, or threatening. What is apparent is that the groups formed around political themes do tend to score higher in the toxic, severely toxic, and insulting categories, with two notable exceptions: the

[19] https://www.perspectiveapi.com/

SandersForPresident subreddit, while clearly political, scores relatively low in those categories, and the *flatearth* subreddit, which would generally not be considered a political topic but rather a satire, scores relatively high. Finally, it is notable that the only standout result in the threatening category is in the *republicans* subreddit, which scores comparatively high. Further analysis of the subreddits in these two groups is certainly warranted. Appendix B-7 contains the Python 3.8 code used to retrieve the toxicity data for the posts for each subreddit from Google's Perspective Comment Analyzer API.

Table 6-5: Minimal and Maximal Subreddit Group Perspective Analysis

TSM	subreddit	tox	sev_tox	insult	threat
64	SandersForPresident	0.145977	0.012856	0.100261	0.017066
111	flatearth	0.210197	0.035424	0.157614	0.019701
247	trump	0.219781	0.033391	0.172795	0.024620
357	science	0.091539	0.005029	0.050391	0.016454
400	globeskepticism	0.091412	0.008832	0.061091	0.010954
800	cars	0.094910	0.007934	0.055654	0.013554
2638	hiking	0.062757	0.005289	0.030742	0.014378
3264	republicans	0.177229	0.011559	0.123996	0.035883
3742	democrats	0.195538	0.020979	0.142528	0.022767
5187	mlb	0.121920	0.012256	0.081324	0.015289
5703	progressive	0.193169	0.020748	0.140391	0.023045
5779	AmericanPolitics	0.207802	0.017407	0.163453	0.016617

7. Conclusions and Future

7.1. Conclusions

The full spectrum and impact of this thesis is difficult to capture. The overall goal of this study was to discover ways to combat the onslaught of fake news plaguing American and global society. That began with framing a model for representing fake news in its many guises. The fake news model proposed here is ambitious and comprehensive. As the research progressed many useful perspectives were gained, and the potential for the model explored.

One key takeaway was the insight that identifying premises is a critical step in determining whether a news article is helpful or hurtful. If the premises of an article disagree with one's understanding of truth, then it is unlikely that any subsequent arguments could be convincing. However, as previously noted, premises could be completely true and factual, wholly false and fabricated, or anywhere between. Thus, premises in themselves can be used for manipulation and misinformation. Unfortunately, premise recognition, and argument mining as a whole, remains a thorny problem. Likewise, furthering the goal of identifying misleading content using misdirection falls in that same category of relying on improvements to argument mining. Both these contributions are significant in theory, but as yet unproven in application.

On the topic of identifying misleading content, recognizing bias and manipulation are much closer to being realizable. While bias identification might rely on argumentation, which wraps back to concerns just raised, it can also be tackled from a linguistic perspective. Sentiment analysis has advanced rapidly in recent years, though it still sometimes struggles with accuracy [170]. Simply identifying the balance of pro and con

statements in an article can highlight the balance of an article. Similarly, recognizing manipulation can rely heavily on linguistic approaches. Thus, these two aspects of the fake news model could be effectively realized and represent ready contributions to the overall fake new model once the hurdle of argument mining is overcome. Since argument identification is not ready for implementation, a more foundational contribution to the fake new problem was sought. This is how we arrived at investigating echo chambers.

While much work remains to demonstrate the full potential of employing a gravity well model to simulate the behavior of social media platforms, our initial results encourage further investigation. Our primary hypothesis, that interest groups within social media platforms can be effectively modeled with a gravity well simulation, was convincingly confirmed for subreddits (interest groups) within the Reddit social media platform by our analysis of the exiting behavior of the agents in our simulation in 6.3.1 and in Table 6-1. It is interesting to note that this approach is thoroughly dynamic in its approach, relying on the effects of a simulation evolving over time to trace the waxing and waning of interest groups within social media platforms.

Further, per our discussion in the Results, Analysis, and Discussion section regarding identifying echo chambers, it can be seen that the ranking and grouping of subreddits by TSM does not equate simply to an ordering of the groups by size, which one might suspect of a gravity-based model, nor does it correspond to the toxicity of the conversations contained within individual subreddits. Thus, we conclude that associating low TSM values with echo chamber propensity has some promise. As previously mentioned, there is currently no standard for conclusively identifying echo chambers, so determining the validity of our findings remains somewhat open to interpretation and subjectivity. All of

which points to several goals for future work, the subject of the next subsection.

7.2. Future Work

An obvious next step for the echo chamber model will be to apply the gravity well simulation to other social media platforms. This will entail fine-tuning the model and its parameters further to enable the incorporation of these additional social media platforms, as well as adjusting the gravitational force thresholds required for entry to and exit from the gravity well. For this study, the entry threshold was set to ensure no agents re-entered the gravity well once they exited. Future work will enable agents to rejoin the gravity well should conditions warrant it. Finally, the universal gravitational constant might also require tuning to allow for a wider spectrum of social media platforms. Throughout these adjustments, the results obtained for Reddit should remain stable between the subreddits, despite their magnitudes shifting to accommodate additional social media platforms.

As previously identified, there is also room for improvement with the calculation of the affinity of an individual for a given group topic. The current calculation makes use of a subreddit's own description, which sometimes is relevant to the content of submissions, but sometimes is only guidance for the participants with little relevance to the actual content of submissions. Additionally, while subreddits have a published group description, there is no guarantee as to their existence or accuracy for other social media platforms. Developing a general measure of affinity between a group's users and the central theme or purpose of the group would enhance the effectiveness of the simulation. This requires a cross-platform analysis of a spectrum of social media platforms to yield a suitable measure of affinity that works for most if not all social media platforms.

Another area for improvement in the gravitational model is to flesh out the definition and use of the tendency for confirmation bias for each user (agent) in the simulation. Confirmation bias has been well-studied in academia [160], [171], [172], but measuring that tendency for individual social media users remains a challenge, though a highly desirable goal.

Finally, an immediate goal is to continue exploring methods for concretely identifying the existence of echo chambers, no matter how laborious, so that we have a method for establishing the efficacy of our proposed echo chamber model. Corresponding with that is the desire to further distinguish the results and findings from this current experiment, particularly with respect to identifying what separates the minimal and maximal TSM groups. One possibility that occurs is to attempt further analysis of text postings for hints of fake news content or influences could prove especially helpful. A return to the spatial aspects of texts yielded by DGs might prove an effective addition to the temporal strengths of the simulation. As our initial investigation into DGs was quite limited, revisiting DGs and additional network analysis could potentially yield insights on the type of content being generated in the low- and high-TSM subreddits, and thus help with distinguishing harmful echo chambers from innocuous ones.

Beyond all those goals for the echo chamber model, there remains the subject of the overarching fake news framework. Much work remains to produce a completely functioning version of that framework. Some of that is a matter of effort and resources, as well as speed, if there is any hope of competing with the rapid advances in fake news generation. Generative artificial intelligence [173], such as that found in OpenAI's

generative pre-trained transformers (GPTs) and ChatGPT[20] [174], and in Google Bard[21], hold a great deal of promise, both for positive efforts and destructive efforts like fake news. Beyond that, some aspects of the framework are still very much in the research stage. Argumentation analysis requires yet more basic research to advance the concept to the point where it might be employed in a production suite. Therefore, a long-term goal for fake news identification is the production of a framework for readily and reliably identifying fake news, and we firmly believe our proposed framework could be a significant contributor to that effort, but a much larger, well-financed effort would be required to turn that proposed framework into reality. Hence, for a more immediate contribution, the gravity well model for echo chambers promises a much speedier path to actionable outcomes.

[20] https://openai.com/blog/chatgpt

[21] https://bard.google.com/

Appendices

Appendix A. Bayesian Knowledge Bases

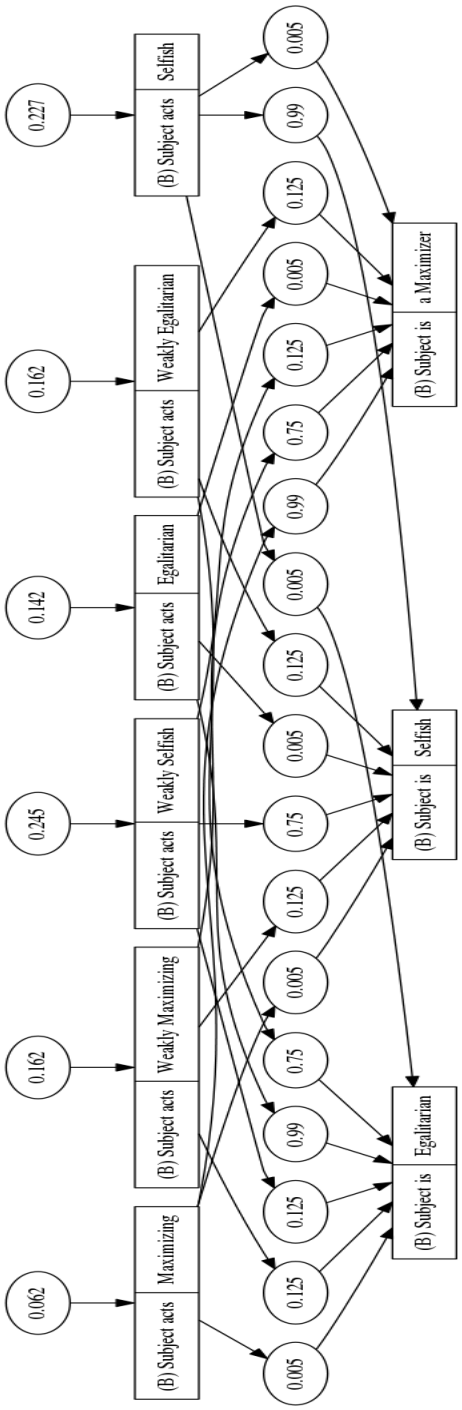

Figure A-1: Initial BKB

When reasoning with BKBs representing a probabilistic complex system, there are typically three different questions to answer that may be the goal of the analysis [98]:

1. What is the most probable state of the world given the evidence?
2. What is the most likely state of an RV given the evidence?
3. What is the most probable composite state of a set of RVs given the evidence?

For the analysis in this thesis, the question to be answered was that of number 2 – what is the probable value of an RV given an initial state of the model updated by player information as time progresses? The updated information in this context was gained by observing player moves over time.

To accomplish this, an initial BKB was built to represent the desired starting conditions, including a first estimate of probabilities for all RVs. Figure A-1 represents one such initial BKB. As can be observed, a BKB consists of

instantiation nodes (I-nodes) and support nodes (S-nodes), connected by directed edges. Each I-node represents a particular state of some random variable. S-nodes may have zero or more incoming edges and only one outgoing edge. Each S-node encodes the if-then rule consisting of these I-nodes and which represents the probability value of the rule.

Once an initial BKB is formed, it can be updated through a process known as belief updating. Belief updating employs Bayes' theorem to perform Bayesian inferencing [175] to incorporate new information as it becomes available. Belief updating can be used to identify probable states and their statistical probabilities. This will change as new information, in this case in the form of an action BKB generated for each player's most recent action, is incorporated into the model using a process called BKB fusion—explained in detail in [176]. For the example at hand, the model is also fused with a strategy identifier BKB, the goal of which was to provide an updated estimate of the current strategy (weak/strong egalitarian, weak/strong selfish, or weak/strong maximizer) and therefore the probable next play for a given player, based on updates to the player's actions. Updates such as these can then be performed for each new action to provide updated predictions of each player's move for the next turn.

Appendix B. Gravity Well Code

Appendix B-1. par_process_pushshift_authors.py

```
'''
Parallel process specified .zst archive file for specified subreddit auths.json
file. Essentially retrieves list of authors from auths.json file to
search for through .zst archive file and retrieve all posts and comments
by those authors. Must call script from commandline with .zst filename and
auths.json filename as parameters.
'''

import zstandard as zstd    # for .zst compressed files
import argparse     # for parsing commandline arguments
import json     # for processing json files
import pandas as pd     # use pandas Python Data Analysis Library
import datetime as dt
import pathlib  # for current execution directory info
from multiprocessing import Pool    # multiprocessor pool
from multiprocessing import cpu_count   # get cpu count
import sys      # for sys.exit

def process_line(line):
#   global posts_df
    obj = json.loads(line)
    # do something with the object here
    if obj['author'] in authors_list:
        # return DF with only desired keys/columns
        return pd.DataFrame([{k: obj[k] for k in keys_to_keep}])

# Begin processing for call from command line with optional output folder name
if __name__ == "__main__":
    if not len(sys.argv) > 1:
        sys.exit("Must call script from commandline with .zst filename and " +
                "auths.json filename as parameters.")

# Use auto python doc description
    my_parser = argparse.ArgumentParser(description=__doc__,
            formatter_class=argparse.RawDescriptionHelpFormatter)
# process zst file parameter
    my_parser.add_argument('zstfile',
        help="specify .zst filename to be decompressed and streamed",
        type=str)
```

```
# process authors json file parameter
    my_parser.add_argument('authsfile',
        help="specify authors json filename for filtering authors",
        type=str)
    my_args = my_parser.parse_args()

    try:
        zstfp = pathlib.Path(my_args.zstfile)
        if not zstfp.exists():
            raise FileNotFoundError(f'zstfile {str(zstfp)} does not exist!')
        authsfp = pathlib.Path(my_args.authsfile)
        if not authsfp.exists():
            raise FileNotFoundError(f'authsfile {str(authsfp)} does not
exist!')
        # Create found author posts json file from file arguments
        json_fp = pathlib.Path(zstfp.parent, authsfp.stem + '_' + zstfp.stem +
'.json')
        json_fp.touch(exist_ok=False)   # create file to mark eventual output
file
    except FileExistsError:
        raise FileExistsError(f'\n\n*** {str(json_fp)} already exists! Exiting
script...')

# columns to be saved from input file
    keys_to_keep = ['author', 'author_created_utc','author_fullname',
            'created_utc','id','permalink','subreddit','subreddit_id']

# create list of authors with whose posts we are concerned
    with open(authsfp, 'r') as af:
       auth_df = pd.DataFrame(json.load(af))
    authors_list = list(auth_df['author'])
    authors_list.remove('[deleted]')  # remove deleted author posts
#authors_list = ["bethanyk98","shingofan","_Renlor"]

# code snippet from Watchful at
##
https://www.reddit.com/r/pushshift/comments/ajmcc0/information_and_code_example
s_on_how_to_use_the/ef012vk/
    obj = None  # placeholder to ensure obj is accessible in iPython
    posts_df = pd.DataFrame()   # create empty DF for incrementally adding
objects
    processes = cpu_count()     # identify nbr of processors available
```

```
    print('Utilizing %d cores' % processes)
    pool = Pool(processes)      # create pool of processes
    with open(zstfp, 'rb') as fh:
        dctx = zstd.ZstdDecompressor(max_window_size=2147483648)
        with dctx.stream_reader(fh) as reader:
            chunk_ctr = 1   # show progress when processing chunks
            previous_line = ""
            print('Processing chunks...' + dt.datetime.now().strftime(
                "%Y/%m/%d %H:%M:%S"))
            while True:
#       if True:    # added to leave indents with 'while' commented out
                chunk = reader.read(2**24)  # 16mb chunks
                if not chunk:
                    break
                print(f'\tProcessing chunk number: {chunk_ctr:10d}',end='\r')
                chunk_ctr += 1
                string_data = chunk.decode('utf-8')
                lines = string_data.split("\n")
                lines[0] = previous_line + lines[0]   # chunk broken in middle
of line

                # don't process last (incomplete) line
                results = pool.map(process_line,lines[:-1])
                posts_df = pd.concat([posts_df]+results,ignore_index=True)

                previous_line = lines[-1]   # chunk may have broken in middle
of line

# Write found author posts to json file
    print(f'\nWriting dataframe to json file {str(json_fp)}.')
    posts_df.to_json(json_fp, orient='records')

    print("\nFinished: " + dt.datetime.now().strftime("%Y/%m/%d %H:%M:%S"))
```

Appendix B-2. par_process_pushshift_agreement.py

```
'''
Parallel process specified month and year .zst comment and submission archive
files for specified subreddit, examining (dis)agreement among author posts and
comments with entire subreddit. Search .zst archive files for subreddit posts
and comments, then evaluate (dis)agreement among all posts for that month for
that subreddit. If json file already exists, reads processed data from file.
Must call script from commandline with year, month, and subreddit name as
parameters. The script expects to run in same directory where files will be
accessed and stored. This script is a precursor to the "sr_sim_ec.py" script.
'''

import par_zst_to_df as pz     # local module for converting zst to dataframe
import praw     # for accessing reddit feeds
import zstandard as zstd    # for .zst compressed files
import argparse     # for parsing commandline arguments
import json     # for processing json files
import pandas as pd     # use pandas Python Data Analysis Library
import datetime as dt
import pathlib  # for current execution directory info
import sys      # for sys.exit and sys.argv to check number of arguments
import numpy as np
from sentence_transformers import SentenceTransformer
from tqdm import tqdm
from sklearn.metrics.pairwise import cosine_similarity  # for similarity calc

def get_sr_description(subreddit,fp):
    ''' Method to retrieve subreddit description from reddit and save to file
'''
    secret = "hK8rMgJ801dpo6sMvpCqKyvwqXDMeA"
    cid = "meUVxGWvFqe9xTaDZsWvsQ"
    user = "EC-desc-scraper"

    # create read-only app instance
    reddit_ro = praw.Reddit(client_id=cid,
            client_secret=secret,
            user_agent=user)
    sr = reddit_ro.subreddit(subreddit)
    fp.write_text(sr.public_description) # write subreddit description to file

def create_std_text_col(posts_df):
# Create text column with strings from either selftext, title, or body
```

```
    # First set text column for all rows to body, as there are usually more
    ## comments than submissions
    if 'body' in posts_df.columns:
        posts_df['text']=posts_df.body
    # now set all rows from submissions to selftext column
    if 'selftext' in posts_df.columns:
        posts_df.loc[posts_df['text'].isna(), 'text'] = posts_df.selftext
    # finally, set all rows with no selftext to title column
    elif 'title' in posts_df.columns:
        posts_df.loc[(posts_df['text']=='') | (posts_df['text']=='[removed]'),
            'text'] = posts_df.title
# Insert entry in first row containing the subreddit description as text
# Will use similarity with description to determine author affinity
    tdf = pd.DataFrame([[subr,subr,subr_desc]], columns=['author',
        'subreddit','text'])
    posts_df = pd.concat([tdf,posts_df]).reset_index(drop=True)

BATCH = 8   # batch size of 8 seems imperically most efficient for intuition
nodes

# Begin processing for call from command line with optional output folder name
if __name__ == "__main__":
    if not len(sys.argv) > 1:
        sys.exit("Must call script from commandline with year, month, and " +
                 "subreddit name as parameters.")
    print("\nBegan: " + dt.datetime.now().strftime("%Y/%m/%d %H:%M:%S"))

# Use auto python doc description
    my_parser = argparse.ArgumentParser(description=__doc__,
            formatter_class=argparse.RawDescriptionHelpFormatter)
# process zst file parameter
    my_parser.add_argument('year',
            help="specify 4-digit year of .zst files to be decompressed and " +
            "streamed",
            type=str)
    my_parser.add_argument('month',
            help="specify 2-digit month of .zst files to be decompressed " +
            "and streamed",
            type=str)
    my_parser.add_argument('subreddit',
            help="specify subreddit name to be analyzed",
            type=str)
```

```
    my_parser.add_argument('-b', '--batch',
        default=BATCH,
        help="specify batch size for BERT sentencetransformer " +
            "(default: '%(default)s')",
        type=int,
        required=False)
    my_args = my_parser.parse_args()
    subr = my_args.subreddit
    batch_sz = my_args.batch
    posts_df = pd.DataFrame()   # create DF for storing posts

    # retrieve subreddit description from file or reddit, as appropriate
    desc_fp = pathlib.Path(subr + '_desc.txt')
    if not desc_fp.is_file():
        get_sr_description(subr,desc_fp)
    subr_desc = desc_fp.read_text()
    print(f'{subr}:\t{subr_desc}\n')

    try:
        # Create author data json file from supplied arguments
        auths_fp = pathlib.Path(subr + '_' + my_args.year + '-' +
                my_args.month + '_auths_agree.json.gz')   # compress file
        auths_fp.touch(exist_ok=False)   # create file to mark output  author
file
    except FileExistsError:
        # if file already exists, issue warning and exit
        print(f"\t*** File {auths_fp} already exists! Move or delete file and
retry. ***")
        raise SystemExit

    try:
        # Create found posts json file from supplied arguments
        agree_fp = pathlib.Path(subr + '_' + my_args.year + '-' +
                my_args.month + '_agree.json.gz')     # compress file
        agree_fp.touch(exist_ok=False)   # create file to mark eventual output
file
    except FileExistsError:
        # if file already exists, read it in rather than parsing zst files
        print(f"\tReading dataframe from {agree_fp}")
        with open(agree_fp, 'rb') as f:  # need 'rb' for zipped files!
            posts_df = pd.read_json(f,compression='gzip')
        print(f'\tRead {posts_df.shape} dataframe from json file {agree_fp}.')
```

```
    else:   # otherwise parse zst files for required data
        print(f"\tCreating dataframe for output to {agree_fp}")
        rs_zstfp = pathlib.Path("RS_"+my_args.year+"-"+my_args.month+".zst")
        if not rs_zstfp.exists():
            raise FileNotFoundError(f'zstfile {rs_zstfp} does not exist!')
        rc_zstfp = pathlib.Path("RC_"+my_args.year+"-"+my_args.month+".zst")
        if not rc_zstfp.exists():
            raise FileNotFoundError(f'zstfile {rc_zstfp} does not exist!')
        print(f'Decompressing data from {rs_zstfp} and {rc_zstfp}.')
        for zstfp in [rs_zstfp,rc_zstfp]:
            if zstfp == rs_zstfp:
                # columns to be saved from RS input file
                keys_to_keep = ['author',
'author_created_utc','author_fullname',
                    'created_utc','id','is_self','permalink','subreddit',
                    'subreddit_id','selftext','title']
            else:
                # columns to be save from RC input file
                keys_to_keep = ['author',
'author_created_utc','author_fullname',
                    'created_utc','id','permalink','subreddit','subreddit_id',
                    'parent_id','body']
            posts_df =
pz.decompress_file_chunks(zstfp,posts_df,[subr],keys_to_keep)
        create_std_text_col(posts_df)
        # drop None text entries
        posts_df.dropna(subset=['text'],inplace=True)
        # Write found subreddit posts to json file
        print(f'\nWriting {posts_df.shape} dataframe to json file
{str(agree_fp)}.')
        posts_df.to_json(agree_fp, orient='records', compression='infer')

    finally:
        # Process posts for input into echo chamber model
        ## i.e. run Bert on all documents to measure similarity
        print("\n\tProcess dataframe for echo chamber model: " +
                dt.datetime.now().strftime("%Y/%m/%d %H:%M:%S"))
        print('jet - exiting for testing...')
        sys.exit()

        documents = posts_df.text
        model_name =  'all-mpnet-base-v2'
```

```
        print(f"\n\tmodel =
SentenceTransformer({model_name}),batch={batch_sz}")
        model = SentenceTransformer(model_name)
        text_embeddings = model.encode(documents, batch_size = batch_sz,
                show_progress_bar = True)
        similarities = cosine_similarity(text_embeddings)
        # change similarity range to [0,1]
        mns = np.min(similarities)     # get bottom of range
        if mns < 0:     # if bottom of range is less than zero
            similarities = similarities - mns
        mxs = np.max(similarities)     # get top of range
        if mxs > 1:     # if top of range is greater than one
            similarities = similarities/mxs
        # Get number of items in upper triangle (minus main diagonal)
        srow_len = np.size(similarities,1)
        # Get number of items in upper triangle (minus main diagonal)
        nbr_sims = srow_len*(srow_len-1)/2
        # Find upper and lower triangles of similarities matrix (minus diag)
        sim_ut = np.triu(similarities,1)
        sim_lt = np.tril(similarities,-1)
        sult = sim_ut + sim_lt  # Create sim array with zero diagonal
        # Calculate the average overall similarity
        s_avg = sim_ut.sum()/nbr_sims
        # Calculate average similarity and dissimilarity for each document
        s_doc_avg = np.array([r.mean() for r in sult])
        d_doc_avg = 1 - s_doc_avg
        # Find all (unique) authors in database
        #authors = list(posts_df[:100].author.unique()) #testing
        authors = list(posts_df.author.unique())
        # Calculate average sim & dissimilarity for each author and store in
dataframe
        auth_df = pd.DataFrame()
        auth_df['author'] = authors
        auth_df['avg_sim'] = [s_doc_avg[posts_df.author[posts_df.author==
            auth].index].mean() for auth in authors]
        auth_df['avg_diss'] = [d_doc_avg[posts_df.author[posts_df.author==
            auth].index].mean() for auth in authors]
        # Try diss to sim ratio for each author and store in DF
        auth_df['diss_sim_ratio'] = auth_df['avg_diss']/auth_df['avg_sim']
        # Use similarity of authors' posts to description to determine affinity
        auth_df['affinity'] = [similarities[0][posts_df.author[posts_df.
            author==auth].index].mean() for auth in authors]
```

```
        # Try diss to affinity ratio for each author and store in DF
        auth_df['diss_affinity_ratio'] =
auth_df['avg_diss']/auth_df['affinity']

        # Write author data to json file for input to sim_ec.py
        print(f'\nWriting {auth_df.shape} dataframe to json file
{str(auths_fp)}.')
        auth_df.to_json(auths_fp, orient='records', compression='infer')

        print("\nFinished: " + dt.datetime.now().strftime("%Y/%m/%d %H:%M:%S"))
```

Appendix B-3. sr_sim_ec.py

```
'''
Gravity well echo chamber simulation for subreddits
e.g. call: %timeit python3 sim_ec.py

Reads json files for simulation parameters, e.g.
******************************************************
{
    "NUMBER_ITERATIONS": 30,
    "TOPIC_SOURCE_MODIFIER": 0,
    "TECHNOLOGY_MODIFIER": 0
}
******************************************************
This script makes required use of files generated by
the par_process_pushshift_agreement.py script.
'''
from jetlogging import * # customized logging module
import argparse # for parsing commandline arguments
from scipy.stats import truncnorm
import json
import random
import math
import numpy as np
import pandas as pd
import datetime as dt
import sys      # for sys.exit and sys.argv to check number of arguments
import pathlib  # for current execution directory info
import csv      # for writing and reading simulation results
import operator as op   # fascilitate passing arithmetic operators as
parameters
from functools import reduce

# Define which attributes from agents file will be used in model
AGENT_ATTRIBUTES =
['author','avg_sim','avg_diss','affinity','diss_affinity_ratio']
class Agent:
    def __init__(self,author,affinity,dissimilarity,bias_seeking=1):
        ''' Agent class for representing members of EC '''
        self._bias_seeking = bias_seeking   # confirmation bias-seeking (1 =
normal/neutral)
        self.captured_ctr(0)   # track how long an agent has been in EC
        self._author = author
```

```
        self._affinity = affinity*bias_seeking
        # dissimilarity is a measure of the dissimilarity of an agent's posts
        ## wrt other EC members's posts
        self._dissimilarity = dissimilarity

    def __init__(self,agt_rec,bias_seeking=1):
        ''' Agent class for representing members of EC '''
        self._bias_seeking = bias_seeking   # confirmation bias-seeking (1 =
normal/neutral)
        self._captured_ctr = 0   # track how long an agent has been in EC
        self._author = agt_rec.Index    # author column has been set to Index
        # affinity is a measure of the similarity of an agent's posts  with the
        ## central theme of EC, including affirmation bias-seeking affinity
        self._affinity=agt_rec.affinity*self._bias_seeking
        '''
        # affinity is a measure of the similarity of an agent's posts
        ## wrt other EC members's posts
        # always use bias_seeking modifier when setting affinity
        self._affinity = agt_rec.avg_sim*self._bias_seeking
        '''
        # dissimilarity is a measure of the dissimilarity of an agent's posts
        ## wrt other EC members's posts
        self._dissimilarity = agt_rec.avg_diss

    @property
    def captured_ctr(self):
        '''Author represents agent's name in the subreddit.'''
        return self._captured_ctr
    @captured_ctr.setter
    def captured_ctr(self,ctr):
        self._capture_ctr = ctr

    @property
    def author(self):
        '''Author represents agent's name in the subreddit.'''
        return self._author
    @author.setter
    def author(self,author):
        self._author = author

    @property
    def affinity(self):
```

```
        '''
        Affinity represents agent's attraction to EC theme, including
        affirmation bias.
        '''
        return self._affinity
    @affinity.setter
    def affinity(self,affinity):
        if affinity < 0.01:
            # Affinity cannot be less than or equal to zero (div by zero)
            affinity = 0.01
        # always use bias_seeking modifier when setting affinity
        self._affinity = affinity*self._bias_seeking

    @property
    def dissimilarity(self):
        '''Dissimilarity represents agent's contrast to others in echo
chamber.'''
        return self._dissimilarity
    @dissimilarity.setter
    def dissimilarity(self,dissimilarity):
        if dissimilarity <= 0.01:
            # Dissimilarity cannot be less than or equal to zero (div by zero)
            dissimilarity = 0.01
        self._dissimilarity = dissimilarity

class EchoChamber:
    ''' EchoChamber represents the EC and the agents captured within '''
    def __init__(self,agents_df,src_mod=1,tech_mod=1,exit_frac=0.1):
        '''
        Generate initial captured agents from provided agents dataframe. Note
        that synthetic model only started with a single captured agent, but
        subreddit model begins with all users already in the subreddit, and no
        "free" agents.
        '''
        # create generator for agents to consider exiting EC
        self.exit_gen = np.random.RandomState()
        # find max affinity, then track as each agent is captured
        self.max_affinity = max(agents_df.affinity)
        self._src_mod = src_mod  # represents pull created by misinformation
sources
        self._tech_mod = tech_mod    # represents pull created by OSN news
feeds
```

```
        self._exit_frac = exit_frac     # fraction of agents to try to leave
        # create list of captured agents, which all of them are to start
        self.captured_agents = [Agent(agent_record)
                for agent_record in agents_df.itertuples()]

    def capture_agent(self,agent):
        ''' Add agent to the captured_agents list '''
        # note: max_affinity never decreases, even if agent with max leaves
        self.max_affinity = max(agent.affinity,self.max_affinity)
        self.captured_agents.append(agent)

    def release_agent(self,agt):
        ''' Release indexed agent from the captured_agents list '''
        #jet self.captured_agents[idx].captured_ctr = 0
        self.captured_agents.remove(agt)

    def update_captured_agents(self):
        '''
        Update affinity for captured agents based on random normal draw ranging
        between current agent affinity minus one sd to max EC affinity, with
        mean set to current agent affinity. Update dissimilarity based on the
        current size of the EC and how long each agent has been captured by the
        echo chamber
        '''
        for agt in self.captured_agents:
            agt.captured_ctr += 1
            # dissimilarity is a randomized function of size of EC
            # use min function here to ensure dissimilarity does not exceed 1
            agt.dissimilarity = min(1,
                self.exit_gen.poisson(agt.dissimilarity*self.mass)/self.mass)
            # in the EC, affinity is further catalyzed by misinformation
            ## <cite> Cinelli, M., De Francisci Morales, G., Galeazzi, A.,
Quattrociocchi,
            ### W., & Starnini, M. (2021). The echo chamber effect on social
media.
            ### Proceedings of the National Academy of Sciences, 118(9),
e2023301118.
            ### https://doi.org/10.1073/pnas.2023301118
            ## <cite> Lyons, B. A., Montgomery, J. M., Guess, A. M., Nyhan, B.,
&
            ### Reifler, J. (2021). Overconfidence in news judgments is
associated with
```

```
            ### false news susceptibility. Proceedings of the National Academy
of Sciences
            ### of the United States of America, 118(23).
            ### https://doi.org/10.1073/pnas.2019527118
            # and group polarization
            ## <cite>Sunstein, C. R. (2005). The Law of Group Polarization.
            ### SSRN Electronic Journal. https://doi.org/10.2139/ssrn.199668
            # Research indicates viewpoints usually become more extreme in ECs
            if (agt.affinity < self.max_affinity):   # max affinity agt doesn't
change
                mean = agt.affinity
                high = min(1,self.max_affinity) # ensure high <= 1
                # We are using the right portion of a normal distribution to
reflect
                ## diminishing likelihood as the bias (affinity) becomes more
                ## extreme, with some probability that the affinity could
decrease
                sd = (high - mean)/2
                low = mean - sd
                # use min to ensure affinity does not exceed 1
                agt.affinity = min(1,truncnorm(
                        a=(low-mean)/sd, b=(high-mean)/sd, loc=mean,
scale=sd).rvs())

    def nbr_exiting(self):
        '''
        nbr_exiting represents the number of members considering exiting
        the EC, ranging from zero to a given fraction of captured members, with
        bias towards zero
        '''
        return abs(random.randint(0,math.ceil(self.EXIT_FRAC*self.mass))-
                random.randint(0,math.floor(self.EXIT_FRAC*self.mass)))

    @property
    def mass(self):
        ''' mass represents size of EC and is number of captured agents '''
        return len(self.captured_agents)

    @property
    def eff_mass(self):
        '''
        eff_mass represents mass of EC multiplied by the social media's
```

```
        tech modifier
        '''
        return self.mass*self._tech_mod

    @property
    def affin_mod(self):
        '''
        Property reflecting EC's ability to affect agent's affinity due to the
        source/topic pillars. Coded to allow for dynamism if desired later
        '''
        return self._src_mod

    @property
    def EXIT_FRAC(self):
        '''
        Fraction of captured agents to figure into determining nbr_exiting
        '''
        return self._exit_frac

    @property
    def capt_DA_ratio(self):
        capt_cnt = self.mass
        if capt_cnt == 0:
            return (0,0)
        else:
            # include affinity modifier for calculations within ec
            return (sum(ag.dissimilarity for ag in
self.captured_agents)/capt_cnt,
                    sum(ag.affinity*self.affin_mod for ag in
self.captured_agents)/capt_cnt)
            ''' jet - simpler implementation ^there^
            return tuple(map(op.truediv,reduce(lambda x,y:
                # include affinity modifier for calculations within ec

(x[0]+y[0],x[1]+y[1]),[(agt.dissimilarity,agt.affinity*self.affin_mod) for
                agt in self.captured_agents],(0,0)),(capt_cnt,capt_cnt)))
            '''


class Simulation:
    ''' Simulation represents the agents and environment outside the EC '''

    def __init__(self, agents_df, n_iterations, grav_const, src_mod,
```

```
        tech_mod, decay_rate, exit_frac, join_frac=0.1):
    self._G = grav_const    # gravitational constant
    # decay rate for dissimilarity and affinity after leaving EC
    self._DECAY_RATE = decay_rate
    self.agents_df = agents_df
    self._NUM_ITERATIONS = n_iterations
    self._JOIN_FRAC = join_frac

    # create EC containing initial captured agents
    self.ec = EchoChamber(self.agents_df,src_mod,tech_mod,exit_frac)

    self.free_agents = []   # no free agents to start
# end of def __init__(self, ...

def liberate_agent(self,agent):
    ''' Add agent to free_agents list '''
    self.free_agents.append(agent)

def remove_agent(self,agent):
    ''' Remove indexed agent from the free_agents list '''
    self.free_agents.remove(agent)

def update_free_agents(self):
    '''
    For free agents, decay dissimilarity, affinity, and captured counter
    based on time away from EC.
    '''
    for agt in self.free_agents:
        if agt.captured_ctr > 0:
            agt.captured_ctr -= 1
        # dissimilarity decays gradually to original value
        orig_diss = self.agents_df.loc[agt.author].avg_diss
        if agt.dissimilarity > orig_diss/self.DECAY_RATE:
            agt.dissimilarity *= self.DECAY_RATE  # decay dissimilarity
        else:
            agt.dissimilarity = orig_diss
        # affinity decays gradually to original value
        orig_aff = self.agents_df.loc[agt.author].affinity
        if agt.affinity > orig_aff/self.DECAY_RATE:
            agt.affinity *= self.DECAY_RATE  # decay affinity
        else:
            agt.affinity = orig_aff
```

```
    def run_sim(self,join_exit_fn,join_force_threshold,exit_force_threshold,
            rnd_exit,rnd_join,tune=-1,DEBUG=False):
        # begin simulation
        random.seed()    # needed to ensure unique RNG for child processes
        if tune>-1:     # if tune parameter provided, can exit early when limit
blown
            logger.debug("Beginning tuning simulation for max of " +
                    f"{self.NUM_ITERATIONS} iterations")
        else:
            logger.debug(f"\nBeginning simulation for {self.NUM_ITERATIONS}
iterations\n")
        pr_format='''{0}\t {1} \t\t{2[0]:0.4f}/{2[1]:0.4f}\t\t {3}\
                \t{4[0]:0.4f}/{4[1]:0.4f}'''
        # prepare csv file for writing results for debugging
        if DEBUG:
            je_f = open(join_exit_fn,'w',newline='')
            je_file_writer = csv.writer(je_f)
            je_file_writer.writerow(
                    ["Epoch","Joined","Joining","Join pct","Avg Join
Force","Exited"
                        ,"Exiting", "Exit pct","Avg Exit Force","EC-size-pct"]
                    )
        for epoch in range(self.NUM_ITERATIONS):
            # periodic output to show progress and trends
            if DEBUG:
                if (epoch%10==0):
                    if (epoch%400==0):
                        logger.trace("Epoch \t#free \t\tavgD/avgA" +
                                "\t\t#captured\t\tavgD/avgA")
                    logger.trace(pr_format.format(epoch,self.freedom,
                        self.free_DA_ratio,self.ec.mass,self.ec.capt_DA_ratio))

            # update time in EC and values for affinity and dissimilarity
            self.ec.update_captured_agents()

            # decay dissimilarity and captured counter based on time away from
EC
            self.update_free_agents()

            # need to select 'nbr_joining' agents to check if actually join
            nbr_joining = self.nbr_joining()
```

```
            j_a_l = []  # list of joining agents
            if rnd_join:    # randomly select joiners
                # Evenly random selection
                j_a_l = random.sample(self.free_agents,nbr_joining)    # joiner
agents
            else:       # select nbr_joing agents with highest affinity
                j_a_l = sorted(self.free_agents,key=lambda a:
                        a.affinity,reverse=True)[:nbr_joining]
            # iterate through the joining agents to see if they join the EC
            nbr_joined = 0
            f_jtot = 0  # use to calculate average joining force for this epoch
            for j_a in j_a_l:
                # use ratio of dissimilarity to affinity (D/A) to influence
prob of agents
                #   being selected to be among the joining agents _considering_
joining EC
                # F = G*((m1*m2)/r^2) to calculate force pulling joining agents
into EC
                #   where G = gravitation constant (set to 1 for convenience)
                #   m1 = 1 (mass of solo joining agent)
                #   m2 = self.ec.eff_mass, nbr of captured agents * science
modifier
                #   r = radius represented by ratio of dissimilarity to
affinity (D/A)
                jf =
self.G*self.ec.eff_mass/(j_a.dissimilarity/j_a.affinity)**2
                f_jtot += jf     # summing the joining forces
                logger.trace("JF = {:.2f}".format(jf))
                if (jf>join_force_threshold) & (self.freedom>1):     # joiner
captured if not last
                    logger.trace(f'joining agent = {j_a.author}, ' +
                            f'eff mass = {self.ec.eff_mass}, ' +
                            f'diss = {j_a.dissimilarity}, '+
                            f'affinity = {j_a.affinity}')
                    logger.trace(f'join force F = {jf:.2f}')
                    self.ec.capture_agent(j_a)
                    self.remove_agent(j_a)
                    nbr_joined += 1
            # now calculate the average joining force
            f_javg = 0 if nbr_joining == 0 else f_jtot/nbr_joining
            # need to select 'nbr_exiting' agents to check if actually exit
            nbr_exiting = self.ec.nbr_exiting()
```

```
            x_a_l = []  # list of exiting agents
            if rnd_exit:    # randomly select exiters
                # Evenly random selection
                x_a_l = random.sample(self.ec.captured_agents,nbr_exiting)    #
exiter agents
            else:       # select nbr_exiting agents with highest dissimilarity
                x_a_l = sorted(self.ec.captured_agents,key=lambda a:
                        a.dissimilarity,reverse=True)[:nbr_exiting]
            # iterate through the exiting agents to see if they join the EC
            nbr_exited = 0
            f_xtot = 0  # use to calculate average exiting force for this epoch
            # Select 'nbr_exiting' agents to check if actually exiting
            for x_a in x_a_l:
                # use ratio of dissimilarity to affinity (D/A) to influence
prob of agents
                #   being selected to be among the joined agents _considering_
exiting EC
                # F = G*((m1*m2)/r^2) to calculate force pulling joining agents
into EC
                #   where G = 1 (gravitation constant)
                #   m1 = 1 (mass of solo joining agent)
                #   m2 = self.ec.mass, representing nbr of captured agents
                #   r = radius represented by ratio of dissimilarity to
affinity (D/A)
                # Introduce an affinity modifier that incorporates the
characteristics of EC
                     ## only in effect within the EC
                xf =
self.G*self.ec.eff_mass/(x_a.dissimilarity/(x_a.affinity*self.ec.affin_mod))**2
                f_xtot += xf     # summing the exiting forces
                logger.trace("XF = {:.2f}".format(xf))
                if (xf<exit_force_threshold) & (self.ec.mass>1):     # exiter
released if not last
                    logger.trace(f'exiting agent = {x_a.author}, ' +
                            f'eff mass = {self.ec.eff_mass}, ' +
                            f'diss = {x_a.dissimilarity}, '+
                            f'affinity = {x_a.affinity}, ' +
                            f'aff*mod = {x_a.affinity*self.ec.affin_mod}')
                    logger.trace(f'exit force F = {xf:.2f}')
                    self.liberate_agent(x_a)
                    self.ec.release_agent(x_a)
                    nbr_exited += 1
```

```
            # now calculate the average exiting force
            f_xavg = 0 if nbr_exiting == 0 else f_xtot/nbr_exiting
            logger.trace("'nbr joined' = {}/{},\t'nbr exited' = {}/{}".format(
                nbr_joined,nbr_joining,nbr_exited,nbr_exiting))
            # write data to csv, while avoiding div by zero
            joined_pct = 0 if nbr_joining == 0 else nbr_joined/nbr_joining
            exited_pct = 0 if nbr_exiting == 0 else nbr_exited/nbr_exiting
            outstr1 =
f'{epoch},{nbr_joined},{nbr_joining},{joined_pct*100:.2f},'
            outstr2 =
f'{f_javg:.3f},{nbr_exited},{nbr_exiting},{exited_pct*100:.2f},'
            outstr3 = f'{f_xavg:.3f},{self.ec.mass/self.NUM_AGENTS*100:.2f}'
            if DEBUG:
                je_file_writer.writerow([outstr1+outstr2+outstr3])
            if tune>-1:     # if tuning, check to exit early
                if len(self.free_agents)>tune:
                    logger.debug(f'\tExiting early after {epoch+1}
iterations.')
                    return epoch+1    # no point in tuning now!
        # print result at end of simulation
        logger.debug("Epoch \t#free \t\tavgD/avgA\t\t#captured\t\tavgD/avgA")
        logger.debug(pr_format.format(self.NUM_ITERATIONS-1,self.freedom,
            self.free_DA_ratio,self.ec.mass,self.ec.capt_DA_ratio))
        logger.debug(f"\n\t\tSimulation complete.")
        if DEBUG:
            je_f.close()
        return epoch+1
    # end of def run_sim(self):

    def nbr_joining(self):
        '''
        nbr_joining represents the number of members considering (re)joining
        the EC, ranging from zero to a given fraction of free members, with
        bias towards zero
        '''
        return abs(random.randint(0,math.floor(self.JOIN_FRAC*self.freedom))-
                random.randint(0,math.floor(self.JOIN_FRAC*self.freedom)))

    @property
    def freedom(self):
        ''' Freedom represents the number of free agents '''
        return len(self.free_agents)
```

```
    @property
    def free_DA_ratio(self):
        free_cnt = self.freedom
        if free_cnt == 0:
            return (0,0)
        else:
            return (sum(ag.dissimilarity for ag in self.free_agents)/free_cnt,
                    sum(ag.affinity for ag in self.free_agents)/free_cnt)
            ''' jet - simpler implementation ^there^
            return tuple(map(op.truediv,reduce(lambda x,y:
                (x[0]+y[0],x[1]+y[1]),[(agt.dissimilarity,agt.affinity) for
                agt in self.free_agents],(0,0)),(free_cnt,free_cnt)))
            '''

    # following properties represent object constants, as there are getters but
no setters
    @property
    def G(self):
        return self._G    # gravitational constant

    @property
    def DECAY_RATE(self):
        return self._DECAY_RATE     # decay rate for affinity and dissimilarity

    @property
    def NUM_AGENTS(self):
        return len(self.agents_df)

    @property
    def NUM_ITERATIONS(self):
        return self._NUM_ITERATIONS

    @property
    def JOIN_FRAC(self):
        return self._JOIN_FRAC
# end class Simulation:

def exit_prog():
    logger.debug("Exiting...")

def get_agents(agfile):
```

```
    '''
    Read simulation agent data.
    '''
    try:    # try reading from uncompressed file first
        with open(agfile, 'r') as f:
            logger.debug(f'Reading agents from {agfile}.')
            # Load only columns desired for agent definition and index by
author
            agents_df = pd.read_json(f)[AGENT_ATTRIBUTES].set_index('author')
    except FileNotFoundError:     # unsuccessful -- try compressed file
        agfile2 = agfile + '.gz'
        logger.debug(f'File {agfile} not found. Trying {agfile2}')
        try:    # try reading from compressed file
            with open(agfile2, 'rb') as f:  # need 'rb' for zipped files!
                logger.debug(f'Reading agents from {agfile}.')
                # Load only columns desired for agent definition and index by
author
                agents_df = pd.read_json(f,compression='gzip')\
                        [AGENT_ATTRIBUTES].set_index('author')
        except FileNotFoundError:
            logger.error(f'Neither {agfile} nor {agfile2} found.')
            raise
    return agents_df

def get_dat(rcfile,DEBUG=False):
    '''
    Read simulation parameters and agent data.
        # "FILE_BASE": filename base used for creating in/output filenames
        # "NUMBER_ITERATIONS": number of iterations for simulation
        # "GRAVITY_CONSTANT": constant set to produce "reasonable" forces
        # "TOPIC_SOURCE_MODIFIER": modifier multiplied to affinity
        ## 0 <= TSM < 1 has dampening effect
        ## TSM = 1 has no effect
        ## TSM > 1 has magnifying effect
        # "TECHNOLOGY_MODIFIER": modifier multiplied to ec's mass
        ## 0 <= TM < 1 has dampening effect -- perhaps correction by social
media
        ## TM = 1 has no effect (neutral)
        ## TM > 1 has magnifying effect -- likely standard mode for most social
media
        # "DECAY_RATE": decay rate for dissimilarity and affinity after leaving
EC
```

```
        # "JOIN_FORCE_THRESHOLD": value gravity force must exceed to join EC
        # "EXIT_FORCE_THRESHOLD": value gravity force must fall below to exit
EC
        # "RANDOM_JOINERS": boolean for whether random or prioritized joiners
used
        # "RANDOM_EXITERS": boolean for whether random or prioritized exiters
used
    '''
    logger.debug("Reading simulation parameters from {}/{}".format(
        pathlib.Path().resolve(),rcfile))
    with open(rcfile,'r') as f:
        parms = json.load(f)
    basefn = parms.get('FILE_BASE')
    # print out all values to record sim settings
    for key,value in parms.items():
        logger.debug(f'{key} = {value}')
    agfile = basefn + '_auths_agree.json'
    agents_df = get_agents(agfile)
    return (parms,agents_df)

if __name__ != "__main__":
    logger = logging.getLogger('jetlogger')

# Begin processing for call from command line with optional json filename
argument
if __name__ == "__main__":
    if not len(sys.argv) > 1:
        sys.exit("Must call script from commandline with rc filename")
    my_parser = argparse.ArgumentParser(description=__doc__, # Use auto python
doc description
        formatter_class=argparse.RawDescriptionHelpFormatter)
    my_parser.add_argument('-d', '--debug',dest='debug',action='store_true',
            help="specify '-d' to enable extra debugging output")
    my_parser.add_argument('-nd','--no-
debug',dest='debug',action='store_false',
            help="specify '-nd' to disable extra debugging output")
    my_parser.set_defaults(debug=False)
    my_parser.add_argument('rcfile',
        help="specify 'run commands' file containing sim parameters ",
        type=str)
    my_args = my_parser.parse_args()
    logger = jetlogger(pathlib.Path(my_args.rcfile).stem+'.log',my_args.debug)
```

```
    logger.debug('Began')
    (parms,agents_df) = get_dat(my_args.rcfile,my_args.debug)
    '''jet - hold off on following change for now
    # Find agents' exiting date (assumed last date means no exit)
    # First, loop through all RC.zst files in reverse date order
    paths = list(pathlib.Path('.').glob('RC*.zst'))
    paths.sort(reverse=True)
    for path in paths:
        # now, for each file, look for last post made by each author
    jet'''

    # Create simulation. Unlike with synthetic data, the subreddit agents
    # all start in the echo chamber.

    # pass parameters (or default) to new sim object
    sim = Simulation(
            agents_df,
            parms.get('NUMBER_ITERATIONS',90),    # defaults to nominal number
            parms.get('GRAVITY_CONSTANT'),
            parms.get('TOPIC_SOURCE_MODIFIER',1),   # one default has no effect
            parms.get('TECHNOLOGY_MODIFIER',1),     # one default has no effect
            parms.get('DECAY_RATE',0.9),     # 0.9 is a slow decay
            parms.get('EXIT_FRAC',0.1),     # 1/10th of capt members try to
leave
            parms.get('JOIN_FRAC',0.1),     # 1/10th of free members try to
join
            )
    sim.run_sim(
            parms.get('FILE_BASE') + '_je.csv',
            parms.get('JOIN_FORCE_THRESHOLD',500),
            parms.get('EXIT_FORCE_THRESHOLD',500),
            parms.get('RANDOM_EXITERS',False),
            parms.get('RANDOM_JOINERS',False),
            -1,     # not tuning
            my_args.debug
            )
    logger.debug('Finished')
    exit_prog()
```

Appendix B-4. sr_tune_sim.py

```
'''
Tuning for gravity well echo chamber simulation for subreddits
'''
from sr_sim_ec import *
import argparse # for parsing commandline arguments
import sys      # for sys.argv to check number of arguments
from jetlogging import * # customized logging module
from multiprocessing import Pool    # multiprocessor pool
from multiprocessing import cpu_count   # get cpu count
from multiprocessing import Queue   # add Queues for comm
from multiprocessing import Manager   # needed to manage queues
import numpy as np  # used for getting mean value
from random import randint  # used to generate random integer
from tqdm import tqdm   # Progress Bar Made Easy
import pathlib  # for current execution directory info
from concurrent.futures import ThreadPoolExecutor, as_completed

def get_agfile(filebase):
    ''' Ensure consistent setting of agents filename '''
    return filebase + '_auths_agree.json'

def cntr(num1,num2):
    ''' split difference between two nums, with rounding '''
    return round((num1+num2)/2)

def tune_tm(subr_fb,tm_Q=None,DEBUG=False):
    TM_NUDGE_LIM = 20
    agents_df = get_agents(get_agfile(subr_fb))
    # Create simulation.
    tm_val = tm_floor = 1    # use tm_val for tech mod; set lower limit to
tuning
    tm_ceil = TM_TUNE_MIN   # set upper limit to tuning
    tm_prev = tm_val    # use prev to track last value
    tm_ctr = 0     # use ctr to track nbr times prev and current are equal
    tm_nudge_ctr = 0     # use nudge_ctr to track nbr times we've nudged
    #tot_cnt = sys.maxsize   # ensure we enter the while loop
    tm_targ = round(TM_TUNE_MOD*len(agents_df))   # reduce exiters according to
mod
```

```
    targ_upper_rng = round(1.05*tm_targ+1)    # upper bound for acceptable
target range
    targ_lower_rng = round(0.9*tm_targ)      # lower bound for acceptable
target range

    #iters = 0
    sim = None
    logger.debug(f'*** Tuning {subr_fb} tm to allow escapes ~' +
            f'{round(TM_TUNE_MOD*100)}% of EC ({len(agents_df)}) = {tm_targ}:
***')
    #return (randint(1,15))   #jet debug
    #while tm_targ<tot_cnt and tm_val<TM_TUNE_MIN:   # exit if tm_val gets too
large
    while True:
        logger.debug(f'\tStarted {subr_fb} loop for tm val = {tm_val} for ' +
            f'~{targ_lower_rng} - {targ_upper_rng} allowed escapes')
        sim = Simulation(
                agents_df,
                NBR_ITERS,
                GC,
                TSM,    # use tsm value from rc file
                tm_val,     # use tm_val in place of tm
                DR,     # decay rate
                X_FRAC,
                J_FRAC,
                )
        sim.run_sim(
                subr_fb + '_je.csv',
                JFT,
                XFT,
                RX,
                RJ,
                targ_upper_rng,    # break early if tuning limit exceeded
                DEBUG
                )
        tot_cnt = len(sim.free_agents)
        logger.debug(f'*** Finished {subr_fb} with {tot_cnt} free agents for '
+
                f'target of {targ_lower_rng} - {targ_upper_rng} on tm val ' +
                f'= {tm_val}')
        if tot_cnt in range(targ_lower_rng,targ_upper_rng):
            break   # good enough
```

```
        elif tm_targ < tot_cnt:   # tm_val too small
            if tm_val < tm_ceil:    # retune with new floor
                tm_floor = tm_val   # reset floor
            elif tm_ceil < TM_TUNE_MIN:     # retune with reset ceiling
                tm_ceil = cntr(tm_ceil,TM_TUNE_MIN)
            else:   # ceiling reached -- no need to tune further
                tm_val = tm_ceil    # ensure tm_val is at ceiling
                break
        else:   # tm_val too large
            if tm_val > tm_floor:   # retune with new ceiling
                tm_ceil = tm_val    # reset ceiling
            elif tm_floor > 1:  # retune with reset floor
                tm_floor = cntr(tm_floor,1)
            else:   # floor too large
                tm_val = tm_floor   # ensure tm_val is at floor
                break
        if tm_nudge_ctr>TM_NUDGE_LIM:     # too many nudges
            break
        tm_val = cntr(tm_floor,tm_ceil)
        if tm_val == tm_prev:  # nudge tm_val to change repetitive outcome
            tm_ctr += 1
            if tm_ctr > 2:
                tm_nudge_ctr += 1
                tm_val += randint(-tm_ctr,tm_ctr)    # nudge
                logger.debug(f'\t\tnudged {subr_fb} tm {tm_nudge_ctr} times')
        else:   # reset ctr
            tm_ctr = 0
            tm_prev = tm_val
    logger.debug(f'*** Tuned {subr_fb} tm to {tm_val} for ' +
            f'target escapes of ~{targ_lower_rng} - {targ_upper_rng} with ' +
            f'actual of {tot_cnt} ***')
    if tm_Q:    # if queue present from parallel processing
        if tm_nudge_ctr>TM_NUDGE_LIM:     # too many nudges
            tm_Q.put((subr_fb,0))    # return zero to indicate failed run
        else:
            tm_Q.put((subr_fb,tm_val))    # add tm_val to queue of results
    if tm_nudge_ctr>TM_NUDGE_LIM:     # too many nudges
        return 0    # return zero to indicate failed run
    else:
        return tm_val
# end of def tune_tm(subr_fb,DEBUG=False):
```

```
def tune_tsm(subr_fb,tuned_tm,tsm_Q=None,DEBUG=False):
    TSM_NUDGE_LIM = 20
    agents_df = get_agents(get_agfile(subr_fb))
    tsm_floor = tsm_ceil = tsm_val = 1    # use tsm_val for topic source mod
    tot_cnt = sys.maxsize   # ensure we enter the while loop
    tsm_prev = tsm_val    # use prev to track last value
    tsm_ctr = 0     # use ctr to track nbr times prev and current are equal
    tsm_nudge_ctr = 0     # use nudge_ctr to track nbr times we've nudged
    tsm_targ = round(TSM_TUNE_MOD*len(agents_df))   # reduce exiters according
to mod
    targ_upper_rng = round(1.02*tsm_targ+1)    # upper bound for acceptable
target range
    targ_lower_rng = round(0.96*tsm_targ)      # lower bound for acceptable
target range
    sim = None
    logger.debug(f'*** Tuning {subr_fb} tsm to allow escapes ~ ' +
            f'{round(TSM_TUNE_MOD*100)}% of EC ({len(agents_df)}) = {tsm_targ}:
***')
    #return (randint(1,15))   #jet debug -- uncomment to skip this calculation
    # for tuning, TSM*LLM sets the maximum times tuning will try to converge
    ## before aborting the tuning effort
    while True:
        logger.debug(f'\tStarted {subr_fb} loop for tsm val = {tsm_val} for ' +
            f'~{targ_lower_rng} - {targ_upper_rng} allowed escapes')
        sim = Simulation(
                agents_df,
                NBR_ITERS,
                GC,
                tsm_val,   # use tsm_val in place of tsm
                tuned_tm,
                DR,  # decay rate
                X_FRAC,
                J_FRAC,
                )
        sim.run_sim(
                subr_fb + '_je.csv',
                JFT,
                XFT,
                RX,
                RJ,
                targ_upper_rng    # allow sim to break early if tuning limit
exceeded
```

```
                )
        tot_cnt = len(sim.free_agents)
        logger.debug(f'\tFinished {subr_fb} with {tot_cnt} free agents for ' +
                f'target of {tsm_targ} on tsm val = {tsm_val}')
        logger.trace(f'tsm_floor={tsm_floor},tsm_ceil={tsm_ceil},' +
                f'tsm_val={tsm_val},targ_low={targ_lower_rng},' +
                f'targ_high={ targ_upper_rng},freed={tot_cnt}')
        if tot_cnt in range(targ_lower_rng,targ_upper_rng+1):
            break   # good enough
        elif tsm_targ < tot_cnt:   # tsm_val too small
            if tsm_val < tsm_ceil:    # retune with new floor
                tsm_floor = tsm_val   # reset floor
            else:
                tsm_ceil *= 10     # retune with reset ceiling
                tsm_floor = tsm_val # raise the floor
        else:   # tsm_val too large
            if tsm_val > tsm_floor:   # retune with new ceiling
                tsm_ceil = tsm_val    # reset ceiling
            elif tsm_floor > 1:  # retune with reset floor
                tsm_floor = cntr(tsm_floor,1)
                tsm_ceil = tsm_val    # reset ceiling
            else:   # tsm floor of '1' too large
                tsm_val = tsm_floor   # ensure tsm_val is at floor
                logger.info(f'\n\t***Unable to tune {subr_fb} tsm small
enough!')
                break
        if tsm_nudge_ctr>TSM_NUDGE_LIM:     # too many nudges
            break
        tsm_val = cntr(tsm_floor,tsm_ceil)
        if tsm_val == tsm_prev:  # nudge tsm_val to change repetitive outcome
            tsm_ctr += 1
            if tsm_ctr > 2:
                tsm_nudge_ctr += 1
                tsm_val += randint(-tsm_ctr,tsm_ctr)    # nudge
                logger.debug(f'\t\tnudged {subr_fb} tsm {tsm_nudge_ctr} times')
        else:   # reset ctr
            tsm_ctr = 0
            tsm_prev = tsm_val
    logger.debug(f'*** Tuned {subr_fb} tsm to {tsm_val} for ' +
            f'target escapes of ~{targ_lower_rng} - {targ_upper_rng} with ' +
            f'actual of {tot_cnt} ***')
    if tsm_Q:    # if queue present from parallel processing
```

```
        if tsm_nudge_ctr>TSM_NUDGE_LIM:     # too many nudges
            tsm_Q.put((subr_fb,0))    # return zero to indicate failed run
        else:
            tsm_Q.put((subr_fb,tsm_val))    # add tsm_val to queue of results
    if tsm_nudge_ctr>TSM_NUDGE_LIM:     # too many nudges
        return 0   # return zero to indicate failed run
    else:
        return tsm_val
# end of def tune_tsm(subr_fb,tuned_tm,DEBUG=False):

if __name__ == "__main__":
    if not len(sys.argv) > 1:
        sys.exit("Must call script from commandline with rc filename")
    my_parser = argparse.ArgumentParser(
        description=__doc__, # Use auto python doc description
        formatter_class=argparse.RawDescriptionHelpFormatter)
    my_parser.add_argument('rcfile',
        help="specify 'run commands' file containing sim parameters ",
        type=str)
    my_args = my_parser.parse_args()
    print("Reading tuning parameters from {}/{}".format(
        pathlib.Path().resolve(),my_args.rcfile))
    with open(my_args.rcfile,'r') as f:
        tune_parms = json.load(f)
    subr_fbs = tune_parms.get("SUBREDDIT_FILEBASES")
    DEBUG = tune_parms.get("DEBUG_TUNE",False)
    PARALLEL = tune_parms.get("PAR_TUNE",True)  # execute in parallel
    LLM = tune_parms.get("LOOP_LIMIT_MULTIPLIER",5) # to limit while loops
    NBR_ITERS = tune_parms.get("NUMBER_ITERATIONS_TUNE", 90)
    GC = tune_parms.get("GRAVITY_CONSTANT_TUNE", 0.00002)
    TM = tune_parms.get('TM_MOD_TUNE',1)     # one default has no effect
    # for tuning, TSM dictates the value used while initially tuning TM
    ## in order to find a common (miniumum) TM value (specific to the
    ## technology--in this case Reddit) to use while tuning TSM for
    ## each specific subreddit
    TSM = tune_parms.get('TSM_MOD_TUNE',1)   # one default has no effect
    DR = tune_parms.get("DECAY_RATE_TUNE", 0.9)
    X_FRAC = tune_parms.get("EXIT_FRAC_TUNE", 0.1)
    J_FRAC = tune_parms.get("JOIN_FRAC_TUNE", 0.1)
    JFT = tune_parms.get("JF_THRESHOLD_TUNE", 120)
    XFT = tune_parms.get("XF_THRESHOLD_TUNE", 20)
    RJ = tune_parms.get("RANDOM_J_TUNE", False)     # defaults to prioritized
```

```
RX = tune_parms.get("RANDOM_X_TUNE", False)     # defaults to prioritized
TM_TUNE_MIN = tune_parms.get("TM_TUNE_MIN",99)     # used for mult tunings
TM_TUNE_MOD = tune_parms.get("TM_TUNE_MOD",0.1)     # default 10%
TSM_TUNE_MOD = tune_parms.get("TSM_TUNE_MOD",0.01)     # default 1%
# use to skip TM tuning and just use val in rc file
TSMONLY = tune_parms.get("TSMONLY", False)
# use to tune TM only and update val in rc file
TMONLY = tune_parms.get("TMONLY", False)
SR = tune_parms.get("SIM_REPS", 3)
COLORS = ['blue','green']

logger = jetlogger(pathlib.Path(my_args.rcfile).stem+'.log',DEBUG)
logger.info(f'\n\tFrom {my_args.rcfile}, tuning {", ".join(subr_fbs)}')
if not TSMONLY: # tune technology modifier
    logger.debug(f'\n\tBegin tuning with current TM MIN = {TM_TUNE_MIN}')
    for key,value in tune_parms.items():
        logger.debug(f'\t{key} = {value}')
    if PARALLEL:     # execute parallel tuning
        # create all processes to run in parallel
        with Pool() as tmpool:
            tmbar = dict()
            for idx,subr in enumerate(subr_fbs):
                # initialize tm_result with empty lists
                tm_result = {subr: [] for subr in subr_fbs}
                # create progress bars for each subreddit
                kwargs = {
                    'total': SR,
                    'unit': 'it',
                    'unit_scale': True,
                    'position': idx,
                    'leave': True,
                    'colour': COLORS[idx%2],
                    'dynamic_ncols': True,
                    'desc': f'TM tuning {subr}'
                    }
                tmbar[subr] = tqdm(**kwargs)
            tm_m = Manager()
            tm_Q = tm_m.Queue()
            # Start load operations and mark each proc with its subreddit
            procs_tm = {tmpool.apply_async(tune_tm,(subr,tm_Q,DEBUG)):
                    subr for subr in subr_fbs for __ in range(SR)}
            # Create a (dynamic) list to iterate over
```

```
                procs_tml = [(p,sr) for p,sr in procs_tm.items()]
                for proc,subr in procs_tml:
                    try:
                        (sr,r) = tm_Q.get()
                        logger.debug(f'TM tune subreddit {sr}, result = {r}')
                        if r:   # if a valid result queued
                            logger.debug(f'TM tune {sr}, {r}')
                            tm_result[sr].append(r)
                            tmbar[sr].update(1)
                            if tmbar[sr].n==SR:     # done with this bar
                                tmbar[sr].close()
                        else:   # requeue the job
                            logger.debug(f'TM tune resubmit {sr}, result =
{r}')
                            p = tmpool.apply_async(tune_tm,(sr,tm_Q,DEBUG))
                            procs_tm[p]=sr
                            procs_tml.append((p,sr))
                    except Exception as e:
                        logger.error(f'tuning tm {sr,r} generated an exception:
{e}')
                #for bar in tmbar: tmbar[bar].close()
        else:   # tune serially
            tm_result = {}
            logger.warning("\n\t***Tuning TM _serially_***")
            for subr in tqdm(subr_fbs,desc='TM subreddits',colour="blue"):
                tm_result[subr] = [tune_tm(subr,DEBUG=DEBUG)
                        for __ in tqdm(range(SR),
                            desc=f'TM reps for {subr}',colour="magenta")]
        tuned_tm = min([min(tml) for tml in tm_result.values()])   # min tm for
all sims
        if tuned_tm < TM_TUNE_MIN:  # update TM_TUNE_MIN in json
            procs_keylist = list(procs_tm)
            logger.info(f'\n\tReplacing TM_TUNE_MIN {TM_TUNE_MIN} with
{tuned_tm} ' +
                f'found for ' + ', '.join({key
                    for key in tm_result for i,j in enumerate(tm_result[key])
                    if j==tuned_tm}))
            tune_parms['TM_TUNE_MIN'] = tuned_tm
            with open(my_args.rcfile,'w') as f:
                json.dump(tune_parms,f,indent=2)    # use indent to make
readable
        else:   # replace with TM_TUNE_MIN
```

```
            tuned_tm = TM_TUNE_MIN
            logger.info(f'\n\tUsing prev technology modifier = {tuned_tm}')
        sys.exit(f'JET - exiting for checking TM={tuned_tm} modification')
    else:   # replace with TM_TUNE_MIN since did not tune TM
        tuned_tm = TM_TUNE_MIN
        logger.info(f'\n\tSkipped tuning TM. Using prev TM = {tuned_tm}')
    if not TMONLY:
        # now tune topic source modifier based on tuned_tm
        if PARALLEL:     # tune in parallel
            with Pool() as tsmpool:
                tsmbar = dict()
                for idx,subr in enumerate(subr_fbs):
                    # initialize tsm_result with empty lists
                    tsm_result = {subr: [] for subr in subr_fbs}
                    # create progress bars for each subreddit
                    kwargs = {
                        'total': SR,
                        'unit': 'it',
                        'unit_scale': True,
                        'position': idx,
                        'leave': True,
                        'colour': COLORS[idx%2],
                        'dynamic_ncols': True,
                        'desc': f'TSM tuning {subr}'
                        }
                    tsmbar[subr] = tqdm(**kwargs)
                tsm_m = Manager()
                tsm_Q = tsm_m.Queue()
                # Start load operations and mark each proc with its subreddit
                procs_tsm = {tsmpool.apply_async(tune_tsm,(subr_fb,tuned_tm,
                    tsm_Q,DEBUG)): subr_fb for subr_fb in subr_fbs for __ in
range(SR)}
                # Create a (dynamic) list to iterate over
                procs_tsml = [(p,sr) for p,sr in procs_tsm.items()]
                for proc,subr in procs_tsml:
                    try:
                        (sr,r) = tsm_Q.get()
                        if r:   # if a valid result queued
                            logger.debug(f'TSM tune subreddit {sr}, result =
{r}')
                            tsm_result[sr].append(r)
                            tsmbar[sr].update(1)
```

```
                            if tsmbar[sr].n==SR:     # done with this bar
                                tsmbar[sr].close()
                        else:   # requeue the job
                            logger.debug(f'TSM tune resubmit {sr}, result =
{r}')
                            p =
tsmpool.apply_async(tune_tsm,(sr,tuned_tm,tsm_Q,DEBUG))
                            procs_tsm[p]=sr
                            procs_tsml.append((p,sr))
                    except Exception as e:
                        logger.error(f'tuning tsm {sr,r} generated an
exception: {e}')
                #for bar in tsmbar: tsmbar[bar].close()
        else:   # tune serially
            tsm_result = {}
            logger.warning("/n/t***Tuning TSM _serially_***")
            for subr_fb in tqdm(subr_fbs,desc='TSM subreddits',colour="green"):
                tsm_result[subr_fb] = [tune_tsm(subr_fb,tuned_tm,DEBUG=DEBUG)
                        for __ in tqdm(range(SR),
                            desc=f'TSM reps for {subr_fb}',colour="cyan")]
    # END OF     if not TMONLY:

# summarize tuning results
    if not TSMONLY:  # tuned technology modifier
        logger.debug(f'*** Tuned tm to allow ~{round(TM_TUNE_MOD*100)}% ' +
                'escapes of EC size: ***')
        logger.debug(f'\ttm_result = {tm_result}')
    if not TMONLY:  # tuned topic source modifier
        logger.debug(f'*** Tuned tsm to ~{round(TSM_TUNE_MOD*100)}% escapes ' +
                'of EC size: ***')
        logger.debug(f'\ttsm_result = {tsm_result}')
        logger.info(f'\n\taverage topic source modifier for:\n\t\t' +
                '\n\t\t'.join(f'{sr} = {round(np.mean(tsm_result[sr]))}'
                    for sr in subr_fbs))
```

Appendix B-5. get_auths_last_posts.py

```
'''
Script to retrieve all authors in first mo/yr provided (hardcoded for now)
for the list of subreddits provided (again hardcoded), then process all
remaining
months in that year, searching for last post made by each of the initial
authors.
After which, the list of authors for each subreddit is sorted by earliest
departure (last date of posting for each author), ranked, then savd to the
specified (hardcoded) file.
'''
import par_zst_to_df as pz     # local module for converting zst to dataframe
import argparse     # for parsing commandline arguments
import json     # for processing json files
import pandas as pd     # use pandas Python Data Analysis Library
import pathlib  # for current execution directory info
from tqdm import tqdm   # Progress Bar Made Easy
import numpy as np  # for NpEncoder class to convert Numpy Ints for JSON export
import gzip     # to enable data export to compressed file
from collections import OrderedDict     # for storing sorted dictionary of
times
from operator import itemgetter     # for sorting the dictionary of author
times

# Extend the JSONEncoder class to handle Numpy values
class NpEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.int64):
            return int(obj)
        if isinstance(obj, np.integer):
            return int(obj)
        if isinstance(obj, np.floating):
            return float(obj)
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return json.JSONEncoder.default(self, obj)

pdf = pd.DataFrame()
auths_filename = 'auths_by_subr.json.gz'
auths_filename2 = 'auths_by_subr-sorted.json.gz'
'''
subrs = ["republicans"]
```

```
'''
subrs =
["SandersForPresident","flatearth","trump","science","cars","Republican",
        "SocialDemocracy","Freethought","travel","math","NeutralPolitics",
        "PoliticalDiscussion","democrats","hiking","republicans","mlb",
        "progressive","AmericanPolitics"]
# keep bare minimum to reduce memory usage
keys_to_keep = ['author','created_utc','id','subreddit']
yr = 2019
mo = 1
zstcf = f'RC_{yr}-{mo:02}.zst'
zstsf = f'RS_{yr}-{mo:02}.zst'

# get data from first month of interest
rs_zstfp = pathlib.Path(zstsf)
if not rs_zstfp.exists():
    raise FileNotFoundError(f'zstfile {rs_zstfp} does not exist!')
rc_zstfp = pathlib.Path(zstcf)
if not rc_zstfp.exists():
    raise FileNotFoundError(f'zstfile {rc_zstfp} does not exist!')
print(f'Decompressing data from {rs_zstfp} and {rc_zstfp}.')
for zstfp in [rs_zstfp,rc_zstfp]:
    pdf = pz.decompress_file_chunks(zstfp,pdf,subrs,keys_to_keep)
# create dictionary of authors occurring in the first month for each subreddit
## store maximum (latest) created_utc date in dict
auths_by_subr = {}
for s in (pbar:=tqdm(subrs,desc='subreddits',colour="blue")):
    pbar.set_postfix_str(s)     # update label of progressbar
    pdf_subr = pdf[pdf.subreddit==s]
    auths_by_subr[s]={a: pdf_subr[pdf_subr.author==a].created_utc.max()
            for a in set(pdf_subr.author)}
# loop over remaining months in year to determine latest date each
## author makes a post
for m in range(mo+1,13):
    pdf = pd.DataFrame() # reset working dataframe
    zstcf = f'RC_{yr}-{m:02}.zst'
    zstsf = f'RS_{yr}-{m:02}.zst'
    rs_zstfp = pathlib.Path(zstsf)
    if not rs_zstfp.exists():
        raise FileNotFoundError(f'zstfile {rs_zstfp} does not exist!')
    rc_zstfp = pathlib.Path(zstcf)
    if not rc_zstfp.exists():
```

```
        raise FileNotFoundError(f'zstfile {rc_zstfp} does not exist!')
    print(f'**Decompressing data from {rs_zstfp} and {rc_zstfp}.')
    for zstfp in [rs_zstfp,rc_zstfp]:
        pdf = pz.decompress_file_chunks(zstfp,pdf,subrs,keys_to_keep)
    for s in (pbar0:=tqdm(subrs,desc='subreddits',colour="blue",position=0)):
        pbar0.set_postfix_str(s)     # update label of progressbar
        pdf_subr = pdf[pdf.subreddit==s]
        # loop over the authors that appear in the new data only if they
        ## were in the original set of authors
        for auth in (pbar1:=tqdm(auths_by_subr[s].keys() & pdf_subr.author,
                desc='authors',colour="green",position=1)):

auths_by_subr[s][auth]=pdf_subr[pdf_subr.author==auth].created_utc.max()
# writing dictionary of author dictionaries to disk
with gzip.open(auths_filename, 'w') as fileout:
    fileout.write(json.dumps(auths_by_subr,cls=NpEncoder).encode('utf-8'))
# read dictionary back in to verify success
with gzip.open(auths_filename, 'rt') as filein:
    auths_by_subr = json.load(filein)
# now sort dictionaries by earliest to latest date of posting, assuming
## earlier dates represent users who have departed the subreddit, at least
## temporarily
for sr,dct in auths_by_subr.items():     # select each subreddit dict of auths
    auths_by_subr[sr] = OrderedDict(sorted(dct.items(), key=itemgetter(1)))
# now rank by order of departure, then save to dict
for sr,dct in auths_by_subr.items():     # select each subreddit dict of auths
    ptim = 0  # track previous time to see if it's same as current
    pauth = ''      # track previous author (dict index)
    ctr = 0     # create rank ctr for author departure order
    for au,dtim in dct.items():  # select each author and latest post in dict
        if dtim != ptim:
            ctr+=1  # only incr ctr if curr timestamp differs from prev
        ptim = dtim
        pauth = au
        auths_by_subr[sr][au]=(dtim,ctr)    # add rank as part of dict
# writing sorted and ranked dictionary of author dictionaries to disk
with gzip.open(auths_filename2, 'w') as fileout:
    fileout.write(json.dumps(auths_by_subr,cls=NpEncoder).encode('utf-8'))
```

Appendix B-6. analyze_exit_order.py

```
'''
Script to compare the order that authors leave (at least temporarily) each
subreddit,
versus the order agents leave the echo chamber simulation of same subreddit.
Code
makes use of the "auths_by_subr-sorted.json.gz" file created by
"get_auths_last_posts.py".
'''
from sr_sim_ec import *
import json     # for processing json files
import pandas as pd     # use pandas Python Data Analysis Library
import pathlib  # for current execution directory info
import shutil   # for file manipulation
from tqdm import tqdm   # Progress Bar Made Easy
import numpy as np  # for NpEncoder class to convert Numpy Ints for JSON export
import gzip     # to enable data export to compressed file
from collections import OrderedDict     # for storing sorted dictionary of
times
from operator import itemgetter     # for sorting the dictionary of author
times
from sklearn.metrics import mean_absolute_percentage_error  # error calc

# Extend the JSONEncoder class to handle Numpy values
class NpEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.int64):
            return int(obj)
        if isinstance(obj, np.integer):
            return int(obj)
        if isinstance(obj, np.floating):
            return float(obj)
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return json.JSONEncoder.default(self, obj)

pdf = pd.DataFrame()
sorted_auths_filename = 'auths_by_subr-sorted.json.gz'
sorted_agents_filename = 'agents_by_subr-sorted.json.gz'
'''
subrs = ["republicans"]     # test data
'''
```

```
subrs =
["SandersForPresident","flatearth","trump","science","cars","Republican",
        "SocialDemocracy","Freethought","travel","math","NeutralPolitics",
        "PoliticalDiscussion","democrats","hiking","republicans","mlb",
        "progressive","AmericanPolitics"]
its = 100
#its = 10    # test data
agents_by_subr = {}     # dict to track agent exit ranking by subreddit
mape_by_subr = {}   # record mean avg pct error of exit rank by subreddit
# read sorted dictionary of exiting authors for each subreddit
with gzip.open(sorted_auths_filename, 'rt') as filein:
    auths_by_subr = json.load(filein)
# simulate repeatedly for each subreddit to determine common agent exiting
order
for sr in subrs:
    rcfile = f'{sr}_2019-01_sim_rc.json'
    debug = False
    agent_dict = dict.fromkeys(list(auths_by_subr[sr].keys()),0)
    logfile = pathlib.Path(pathlib.Path(rcfile).stem+'.rank.log')
    logger = jetlogger(logfile,debug)
    for it in (pbar:=tqdm(range(its),desc=sr,colour="blue")):
        #pbar0.set_postfix_str(s)     # update label of progressbar
    #for it in range(its):   # repeat for {its} iterations
        logger.debug('Began')
        (parms,agents_df) = get_dat(rcfile,debug)
        parms['TOPIC_SOURCE_MODIFIER'] = 1  # override TSM
        logger.debug(f"overrode TOPIC_SOURCE_MODIFIER =
{parms.get('TOPIC_SOURCE_MODIFIER')}")
        parms['TECHNOLOGY_MODIFIER'] = 1  # override TM
        logger.debug(f"overrode TECHNOLOGY_MODIFIER =
{parms.get('TECHNOLOGY_MODIFIER')}")
        parms['JOIN_FRAC'] = 0.0  #override join fraction
        logger.debug(f"overrode JOIN_FRAC = {parms.get('JOIN_FRAC')}")
        # pass parameters (or default) to new sim object
        sim = Simulation(
                agents_df,
                parms.get('NUMBER_ITERATIONS',90),    # defaults to nominal
number
                parms.get('GRAVITY_CONSTANT'),
                parms.get('TOPIC_SOURCE_MODIFIER',1),   # 1 ensures agents will
exit
```

```
                parms.get('TECHNOLOGY_MODIFIER',1),     # 1 ensures agents will
exit
                parms.get('DECAY_RATE',0.9),     # 0.9 is a slow decay
                parms.get('EXIT_FRAC',0.1),     # 1/10th of capt members try to
leave
                parms.get('JOIN_FRAC',0.1),     # JOIN_FRAC=0 ensures no agents
rejoin
                )
        # run simulation
        sim.run_sim(
                parms.get('FILE_BASE') + '_je.csv',
                parms.get('JOIN_FORCE_THRESHOLD',500),
                parms.get('EXIT_FORCE_THRESHOLD',500),
                parms.get('RANDOM_EXITERS',False),
                parms.get('RANDOM_JOINERS',False),
                -1,     # not tuning
                debug
                )
        logger.debug(f'Finished run {it} of sr_sim_ec for {sr}')
        for idx,agt in enumerate(sim.free_agents):  # free_agents ordered by
exit
            agent_dict[agt.author] += idx+1     # update total exit rank
        # should only be a few agents left in the EC, so they will have highest
rank
        max_rank = len(sim.free_agents)+1
        logger.debug(f'Setting remaining {len(sim.ec.captured_agents)} ' +
                f'captured agents to max rank of {max_rank}')
        for agt in sim.ec.captured_agents:
            agent_dict[agt.author] += max_rank
    agent_dict.update((k,rank/its) for k,rank in agent_dict.items()) # avg rank
    # now sort dictionary by earliest to latest average exit order
    agent_dict = OrderedDict(sorted(agent_dict.items(), key=itemgetter(1)))
    # now rank by order of departure, then save to dict
    prnk = 0  # track previous rank to see if it's same as current
    ctr = 0     # create rank ctr for author departure order
    for agt,rnk in agent_dict.items():  # select each author and avg rank in
dict
        if rnk != prnk:
            ctr+=1  # only incr ctr if curr rank differs from prev
        prnk = rnk
        agent_dict[agt]=(rnk,ctr)    # add rank ctr as part of dict
    # calculate mean absolute error of rankings
```

```
    ## list of real exit rankings
    y_true = [itemgetter(1)(item) for item in list(auths_by_subr[sr].values())]
    ## list of simulation exit rankings
    y_pred = [itemgetter(1)(agent_dict[agt]) for agt in auths_by_subr[sr]]
    mape_by_subr[sr] = mean_absolute_percentage_error(y_true, y_pred)
    logger.debug(f'{sr} ranking MAPE = {mape_by_subr[sr]}')
    print(f'{sr} ranking MAPE = {mape_by_subr[sr]}')
    # close logging for this subreddit
    logs = list(logger.handlers)
    for l in logs:
        logger.removeHandler(l)
        l.flush()
        l.close()
    # compress logfile then delete orig
    with open(logfile, 'rb') as f_in:
        with gzip.open(logfile.name+'.gz', 'wb') as f_out:
            shutil.copyfileobj(f_in, f_out)
    pathlib.Path.unlink(logfile)    # delete orig logfile
    agents_by_subr[sr] = agent_dict     # preserve agent_dict for this
subreddit
# writing sorted and ranked dictionary of agent dictionaries to disk
with gzip.open(sorted_agents_filename, 'w') as fileout:
    fileout.write(json.dumps(agents_by_subr,cls=NpEncoder).encode('utf-8'))
```

Appendix B-7. sr_get_toxicity.py

```
'''
Calculate toxicity using the Google API Perspective Comment Analyzer.
Reads Google API key from "tox_key.txt" file in current directory
'''
from googleapiclient import discovery
import json     # for json files
import pandas as pd     # for dataframes
import numpy as np  # for np.nan
import pathlib  # for path/file specification and manipulation
import argparse     # for parsing commandline arguments
import datetime     # for throttling Google App requests
from time import sleep     # time.sleep for throttling
from tqdm import tqdm   # Progress Bar Made Easy
import sys  # for sys.exit and sys.argv
from ratelimit import limits, sleep_and_retry

# Define variables to keep track of throttling
# 30 calls per minute
CALLS = 60  # number of calls per minute
RATE_LIMIT = 60     # time to wait when calls exceeded
TXT_LIMIT = 20480   # text length limit for Google API query

@sleep_and_retry
@limits(calls=CALLS, period=RATE_LIMIT)
def check_limit():
    ''' Empty function just to check for calls to API '''
    return

@sleep_and_retry
@limits(calls=CALLS, period=RATE_LIMIT)
def get_post_attr(txt: str) -> (float,float,float):
    ''' analyze text for toxicity '''
    analyze_request = {
            'comment': { 'text': txt},
            'languages':[lang],
            'requestedAttributes': {
                'TOXICITY': {},
                'SEVERE_TOXICITY': {},
                'INSULT': {},
                'THREAT': {},
                }
```

```
            }
    # get the attributes for this text
    response = client.comments().analyze(body=analyze_request).execute()
    # parse the results
    tox=response['attributeScores']['TOXICITY']['summaryScore']['value']

sev_tox=response['attributeScores']['SEVERE_TOXICITY']['summaryScore']['value']
    insult=response['attributeScores']['INSULT']['summaryScore']['value']
    threat=response['attributeScores']['THREAT']['summaryScore']['value']
    return (tox,sev_tox,insult,threat)

with open('tox_key.txt') as f:
    API_KEY = f.read()
lang = "en"

if __name__ == "__main__":
    if not len(sys.argv) > 1:
        sys.exit("Must call script from commandline with filename as
parameter.")

    else:
# Use auto python doc description
        my_parser = argparse.ArgumentParser(description=__doc__,
            formatter_class=argparse.RawDescriptionHelpFormatter)
        my_parser.add_argument('filename',
            help="specify 'filename' from which to retrieve posts",
            type=str)
        my_parser.add_argument('-t', '--test',dest='test',action='store_true')
        my_parser.add_argument('-nt','--no-
test',dest='test',action='store_false')
        my_parser.set_defaults(test=False)
        my_args = my_parser.parse_args()

        fp = pathlib.Path(my_args.filename)
        with open(fp, 'rb') as f:  # need 'rb' for zipped files!
            posts_df = pd.read_json(f,compression='gzip')
            print(f'\tRead {posts_df.shape} dataframe from json file {fp}.')
        if my_args.test:    # truncate posts_df for testing
            posts_df = posts_df[:100]
# Limit text column to limit required by Google API
        posts_df['text']=posts_df['text'].str.slice(0,TXT_LIMIT)
        new_cols = ['tox','sev_tox','insult','threat']
```

```
        posts_df[new_cols] = np.nan     # add new columns

# create client connection to Google API Perspective library
        client = discovery.build(
                "commentanalyzer",
                "v1alpha1",
                developerKey=API_KEY,
                discoveryServiceUrl="https://commentanalyzer.googleapis.com/" +
\
                        "$discovery/rest?version=v1alpha1",
                static_discovery=False,
        )
# calculate toxicity for every posting in dataframe
        tqdm.pandas(desc='Get toxicity',colour='blue')   # construct progress
bar
# use progress_apply from tqdm to show progress
        posts_df.update(posts_df['text'].progress_apply(lambda txt:
                pd.Series(dict(zip(new_cols,get_post_attr(txt))))))
# write updated dataframe to file
        if my_args.test:    # write test results to another file
            fp = pathlib.Path('test_tox.json.gz')
        posts_df.to_json(fp, orient='records', compression='infer')
```

References

1. H. Allcott and M. Gentzkow, “Social Media and Fake News in the 2016 Election,” *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–236, May 2017, doi: 10.1257/jep.31.2.211.

2. A. Guess, B. Nyhan, and J. Reifler, “Selective Exposure to Misinformation: Evidence from the consumption of fake news during the 2016 US presidential campaign,” *European Research Council*, 2018, Accessed: Dec. 02, 2018. [Online]. Available: http://www.ask-force.org/web/Fundamentalists/Guess-Selective-Exposure-to-Misinformation-Evidence-Presidential-Campaign-2018.pdf

3. D. M. J. Lazer *et al.*, “The science of fake news,” *Science (1979)*, vol. 359, no. 6380, pp. 1094–1096, 2018, doi: 10.1126/science.aao2998.

4. J. L. Hochschild and K. L. Einstein, “Do Facts Matter? Information and Misinformation in American Politics,” *Polit Sci Q*, vol. 130, no. 4, pp. 585–624, Dec. 2015, doi: 10.1002/polq.12398.

5. M. Balmas, “When Fake News Becomes Real: Combined Exposure to Multiple News Sources and Political Attitudes of Inefficacy, Alienation, and Cynicism,” *Communic Res*, 2014, doi: 10.1177/0093650212453600.

6. M. Hosenball, “Russia used social media for widespread meddling in U.S. politics: reports | Reuters.” Accessed: Dec. 17, 2018. [Online]. Available: https://www.reuters.com/article/us-usa-trump-russia-socialmedia-idUSKBN1OG257

7. J. Farkas and J. Schou, *Post-Truth, fake news and democracy: Mapping the politics of falsehood*. 2019. doi: 10.4324/9780429317347.

8. H. Wasserman, “Fake news from Africa: Panics, politics and paradigms,” *Journalism*, vol. 21, no. 1, 2020, doi: 10.1177/1464884917746861.

9. R. Rogers and S. Niederer, “The politics of social media manipulation,” in *The Politics of Social Media Manipulation*, 2020. doi: 10.2307/j.ctv1b0fvs5.3.

10. J. L. Pasley, *The Tyranny of Printers: Newspaper Politics in the Early American Republic*. in Jeffersonian America. University of Virginia Press, 2002. [Online]. Available: https://books.google.com/books?id=8xVMxjD0bv4C

11. N. Faulkner, “BBC - History - Ancient History in depth: The Official Truth: Propaganda in the Roman Empire,” BBC History. Accessed: Nov. 23, 2018. [Online]. Available: http://www.bbc.co.uk/history/ancient/romans/romanpropaganda_article_01.shtml

12. L. Sydell, “We Tracked Down A Fake-News Creator In The Suburbs. Here’s What We Learned,” NPR All Tech Considered. Accessed: Dec. 02, 2018. [Online]. Available: https://www.npr.org/sections/alltechconsidered/2016/11/23/503146770/npr-finds-the-head-of-a-covert-fake-news-operation-in-the-suburbs

13. M. Fisher, J. W. Cox, and P. Hermann, “Pizzagate: From rumor, to hashtag, to gunfire in D.C. - The Washington Post,” The Washington Post. Accessed: Nov. 23, 2018. [Online]. Available: https://www.washingtonpost.com/local/pizzagate-from-rumor-to-hashtag-to-gunfire-in-dc/2016/12/06/4c7def50-bbd4-11e6-94ac-3d324840106c_story.html

14. A. Chen, “The Agency,” The New York Times. Accessed: Nov. 23, 2018. [Online]. Available: https://www.nytimes.com/2015/06/07/magazine/the-agency.html

15. L. Frayer, “How The Spread Of Fake Stories In India Has Led To Violence : NPR,” NPR. Accessed: Nov. 23, 2018. [Online]. Available: https://www.npr.org/2018/07/17/629896525/how-the-spread-of-fake-stories-in-india-has-led-to-violence

16. Y. Adegoke and BBC Africa Eye, “Nigerian police say ‘fake news’ on Facebook is killing people - BBC News,” BBC News. Accessed: Nov. 23, 2018. [Online]. Available: https://www.bbc.co.uk/news/resources/idt-sh/nigeria_fake_news

17. J. Roozenbeek *et al.*, “Susceptibility to misinformation about COVID-19 around the world: Susceptibility to COVID misinformation,” *R Soc Open Sci*, vol. 7, no. 10, Oct. 2020, doi: 10.1098/RSOS.201199.

18. J. W. Ayers *et al.*, “Spread of Misinformation About Face Masks and COVID-19 by Automated Software on Facebook.,” *JAMA Intern Med*, Jun. 2021, doi: 10.1001/jamainternmed.2021.2498.

19. Y. M. Rocha, G. A. de Moura, G. A. Desidério, C. H. de Oliveira, F. D. Lourenço, and L. D. de Figueiredo Nicolete, “The impact of fake news on social media and its influence on health during the COVID-19 pandemic: a systematic review,” *Journal of Public Health (Germany)*. 2021. doi: 10.1007/s10389-021-01658-z.

20. C. Timberg and T. Romm, “Facebook CEO Mark Zuckerberg to Capitol Hill: ‘It was my mistake, and I’m sorry.,’” The Washngton Post. [Online]. Available: https://www.washingtonpost.com/news/the-switch/wp/2018/04/09/facebook-chief-executive-mark-zuckerberg-to-captiol-hill-it-was-my-mistake-and-im-sorry/?utm_term=.3c7bade6b6be

21. A. Guess, J. Nagler, and J. Tucker, “Less than you think: Prevalence and predictors of fake news dissemination on Facebook,” *Asian-Australas J Anim Sci*, vol. 32, no. 2, 2019, doi: 10.1126/sciadv.aau4586.

22. V. Le Pochat, L. Edelson, T. Van Goethem, W. Joosen, D. McCoy, and T. Lauinger, “An Audit of Facebook’s Political Ad Policy Enforcement,” in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 607–624.

23. M. Field, “YouTube will use Wikipedia to tackle fake news - but failed to tell Wikipedia.” Accessed: Nov. 22, 2018. [Online]. Available: https://www.telegraph.co.uk/technology/2018/03/15/youtube-will-use-wikipedia-tackle-fake-news-failed-tell-wikipedia/

24. BBC News Staff and R. Cellan-Jones, “California fire conspiracies a hit on YouTube,” BBC News. [Online]. Available: https://www.bbc.com/news/technology-46304972

25. D. Milmo, “YouTube is major conduit of fake news, factcheckers say | YouTube | The Guardian,” *The Guardian*, Jan. 12, 2022. Accessed: Jan. 27, 2023. [Online]. Available: https://www.theguardian.com/technology/2022/jan/12/youtube-is-major-conduit-of-fake-news-factcheckers-say

26. A. Huxley, *Brave New World Revisited*. Harper, 1958. [Online]. Available: https://books.google.com/books?id=d7sOAQAAIAAJ&dq=editions:ISBN0795311699&lr=

27. T. Honderich and M. Ruse, “Reductionism,” *The Oxford companion to philosophy*. Oxford University Press, p. 793, 2005.

28. K. H. Jamieson and J. N. Cappella, *Echo chamber: Rush Limbaugh and the conservative media establishment*. Oxford University Press, 2008.

29. M. Cinelli, G. De Francisci Morales, A. Galeazzi, W. Quattrociocchi, and M. Starnini, “The echo chamber effect on social media,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 9, p. e2023301118, Mar. 2021, doi: 10.1073/pnas.2023301118.

30. J. E. Thompson and E. Santos, “Echo Chambers as Gravity Wells,” in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, May 2023, pp. 848–857. doi: 10.1109/IPDPSW59300.2023.00141.

31. K. Than, “Einstein Theories Confirmed by NASA Gravity Probe,” National Geographic News. Accessed: Jan. 01, 2023. [Online]. Available: https://www.nationalgeographic.com/science/article/110505-einstein-theories-confirmed-gravity-probe-nasa-space-science?loggedin=true&rnd=1672690762546

32. N. Newman, R. Fletcher, A. Kalogeropoulos, D. A. L. Levy, and R. K. Nielsen, "Reuters Institute digital news report 2017," 2017.

33. R. Marchi, "With Facebook, blogs, and fake news, teens reject journalistic 'objectivity,'" *Journal of Communication Inquiry*, 2012, doi: 10.1177/0196859912458700.

34. P. J. Boczkowski and E. Mitchelstein, *The news gap: When the information preferences of the media and the public diverge*. MIT press, 2013.

35. The Central Intelligence Agency, The Federal Bureau of Investigation, and The National Security Agency, "Background to Assessing Russian Activities and Intentions in Recent US Elections: The Analytic Process and Cyber Incident Attribution," CreateSpace Independent Publishing Platform, 2017.

36. V. L. Rubin, Y. Chen, and N. J. Conroy, "Deception detection for news: Three types of fakes," *Proceedings of the Association for Information Science and Technology*, 2015, doi: 10.1002/pra2.2015.145052010083.

37. E. C. Tandoc Jr., Z. W. Lim, and R. Ling, "Defining 'Fake News,'" *Digital Journalism*, vol. 6, no. 2, pp. 137–153, Feb. 2018, doi: 10.1080/21670811.2017.1360143.

38. J. Weedon, W. Nuland, and A. Stamos, "Information Operations and Facebook," 2017. doi: 10.1016/B978-1-4377-2003-7.00058-3.

39. K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake News Detection on Social Media: A Data Mining Perspective," *ACM SIGKDD Explorations Newsletter*, 2017, doi: 10.1145/3137597.3137600.

40. P. J. Shoemaker, M. Eichholz, E. Kim, and B. Wrigley, "Individual and routine forces in gatekeeping," *Journalism and Mass Communication Quaterly*, 2001, doi: 10.1177/107769900107800202.

41. D. M. White, "The 'Gate Keeper': A Case Study in the Selection of News," *Journalism Bulletin*, 1950, doi: 10.1177/107769905002700403.

42. A. Calcutt and P. Hammond, *Journalism studies: A critical introduction*. Routledge, 2011.

43. P. M. Taylor, *British propaganda in the 20th century: Selling democracy*. Edinburgh: Edinburgh University Press, 1999.

44. C. Timberg and T. Romm, "Facebook CEO Mark Zuckerberg to Capitol Hill: 'It was my mistake, and I'm sorry.,'" The Washngton Post. [Online]. Available: https://www.washingtonpost.com/news/the-switch/wp/2018/04/09/facebook-

chief-executive-mark-zuckerberg-to-captiol-hill-it-was-my-mistake-and-im-sorry/?utm_term=.3c7bade6b6be

45. L. Carvalho, “The Case Against Fake News Gatekeeping by Social Networks,” 2017.

46. M. Broersma, “A refractured paradigm: Journalism, hoaxes and the challenge of trust,” in *Rethinking Journalism: Trust and Participation in a Transformed News Landscape*, 2013. doi: 10.4324/9780203102688.

47. G. Muhlmann, *Political history of journalism*. Polity, 2008.

48. I. Gaber, “Three cheers for subjectivity: Or the crumbling of the seven pillars of traditional journalistic wisdom,” *Communications Law*, 2009, doi: 10.1039/c2sm26575c.

49. K. Bunton, “Just the Facts: How Objectivity Came to Define American Journalism,” *American Journalism*, 2001, doi: 10.1080/08821127.2001.10739300.

50. E. Tacchini, G. Ballarin, M. L. Della Vedova, S. Moret, and L. de Alfaro, “Some like it hoax: Automated fake news detection in social networks,” *arXiv preprint arXiv:1704.07506*, 2017.

51. BBC News Staff and R. Cellan-Jones, “California fire conspiracies a hit on YouTube,” BBC News. [Online]. Available: https://www.bbc.com/news/technology-46304972

52. H. Berghel, “Lies, Damn lies, and fake news,” *Computer (Long Beach Calif)*, 2017, doi: 10.1109/MC.2017.56.

53. C. Silverman, “This analysis shows how fake election news stories outperformed real news on Facebook. BuzzFeed, Nov. 16,” BuzzFeed News. Accessed: Dec. 03, 2018. [Online]. Available: https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook

54. S. Wineburg, S. McGrew, J. Breakstone, and T. Ortega, “Evaluating information: The cornerstone of civic online reasoning,” *Stanford Digital Repository. Retrieved January*, vol. 8, p. 2018, 2016.

55. Y. Chen, N. J. Conroy, and V. L. Rubin, “Misleading Online Content: Recognizing Clickbait as ``False News’’,” in *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, 2015. doi: 10.1145/2823465.2823467.

56. B. D. Horne and S. Adali, “This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news,” *arXiv preprint arXiv:1703.09398*, 2017.

57. N. J. Conroy, V. L. Rubin, and Y. Chen, “Automatic deception detection: Methods for finding fake news,” *Proceedings of the Association for Information Science and Technology*, 2015, doi: 10.1002/pra2.2015.145052010082.

58. X. L. Dong *et al.*, “Knowledge-based trust: Estimating the trustworthiness of web sources,” *Proceedings of the VLDB Endowment*, vol. 8, no. 9, pp. 938–949, 2016.

59. C. Timberg, “Russian propaganda effort helped spread ‘fake news’ during election, experts say,” The Washington Post. Accessed: Dec. 08, 2018. [Online]. Available: https://www.washingtonpost.com/business/economy/russian-propaganda-effort-helped-spread-fake-news-during-election-experts-say/2016/11/24/793903b6-8a40-4ca9-b712-716af66098fe_story.html?utm_term=.14843fd3ab0a

60. W. Y. Wang, “‘Liar, Liar Pants on Fire’: A new benchmark dataset for fake news detection,” *arXiv preprint arXiv:1705.00648*, 2017.

61. B. Upadhayay and V. Behzadan, “Sentimental LIAR: Extended Corpus and Deep Learning Models for Fake Claim Classification,” in *Proceedings - 2020 IEEE International Conference on Intelligence and Security Informatics, ISI 2020*, 2020. doi: 10.1109/ISI49825.2020.9280528.

62. H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi, “Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017. doi: 10.18653/v1/D17-1317.

63. S. Volkova, K. Shaffer, J. Y. Jang, and N. Hodas, “Separating Facts from Fiction: Linguistic Models to Classify Suspicious and Trusted News Posts on Twitter,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017. doi: 10.18653/v1/P17-2102.

64. Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

65. J. H. Lau and T. Baldwin, “An empirical evaluation of doc2vec with practical insights into document embedding generation,” *arXiv preprint arXiv:1607.05368*, 2016.

66. E. Gilbert, T. Bergstrom, and K. Karahalios, “Blogs are echo chambers: Blogs are echo chambers,” in *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 2009. doi: 10.1109/HICSS.2009.91.

67. A. Ross Arguedas, C. Robertson, R. Fletcher, and R. Nielsen, “Echo chambers, filter bubbles, and polarisation: a literature review,” 2022.

68. S. Du and S. Gregory, “The echo chamber effect in twitter: Does community polarization increase?,” *Studies in Computational Intelligence*, vol. 693, 2017, doi: 10.1007/978-3-319-50901-3_30.

69. M. Aida and A. Hashizume, “Modeling of Online Echo-Chamber Effect Based on the Concept of Spontaneous Symmetry Breaking,” in *IEICE Proceeding Series, 63*, 2020. [Online]. Available: https://arxiv.org/abs/2011.13372

70. N. Yusuf, N. Al-Banawi, and H. A. R. Al-Imam, “The Social Media As Echo Chamber: The Digital Impact,” *Journal of Business & Economics Research (JBER)*, vol. 12, no. 1, 2013, doi: 10.19030/jber.v12i1.8369.

71. N. Gillani, A. Yuan, M. Saveski, S. Vosoughi, and D. Roy, “Me, my echo chamber, and i: Introspection on social media polarization,” in *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*, 2018. doi: 10.1145/3178876.3186130.

72. L. Harris and P. Harrigan, “Social Media in Politics: The Ultimate Voter Engagement Tool or Simply an Echo Chamber?,” *Journal of Political Marketing*, vol. 14, no. 3, 2015, doi: 10.1080/15377857.2012.693059.

73. G. De Francisci Morales, C. Monti, and M. Starnini, “No echo in the chambers of political interactions on Reddit,” *Sci Rep*, vol. 11, no. 1, pp. 1–12, 2021.

74. L. Terren and R. Borge, “Echo Chambers on Social Media: A Systematic Review of the Literature,” *Review of Communication Research*, vol. 9, 2021, doi: 10.12840/ISSN.2255-4165.028.

75. F. H. Calderón, L.-K. Cheng, M.-J. Lin, Y.-H. Huang, and Y.-S. Chen, “Content-based echo chamber detection on social media platforms,” in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 597–600.

76. P. Törnberg, “Echo chambers and viral misinformation: Modeling fake news as complex contagion,” *PLoS One*, vol. 13, no. 9, 2018, doi: 10.1371/journal.pone.0203958.

77. V. Morini, L. Pollacci, and G. Rossetti, “Toward a standard approach for echo chamber detection: Reddit case study,” *Applied Sciences (Switzerland)*, vol. 11, no. 12, 2021, doi: 10.3390/app11125390.

78. W. H. Dutton, B. C. Reisdorf, E. Dubois, and G. Blank, “Social Shaping of the Politics of Internet Search and Networking: Moving Beyond Filter Bubbles, Echo

Chambers, and Fake News," *SSRN Electronic Journal*, 2017, doi: 10.2139/ssrn.2944191.

79. A. Efstratiou, J. Blackburn, T. Caulfield, G. Stringhini, S. Zannettou, and E. De Cristofaro, "Non-Polar Opposites: Analyzing the Relationship Between Echo Chambers and Hostile Intergroup Interactions on Reddit," in *ICWSM 2023*, 2023. [Online]. Available: https://arxiv.org/abs/2211.14388

80. G. Villa, G. Pasi, and M. Viviani, "Echo chamber detection and analysis: A topology- and content-based approach in the COVID-19 scenario," *Soc Netw Anal Min*, vol. 11, no. 1, 2021, doi: 10.1007/s13278-021-00779-3.

81. F. Baumann, P. Lorenz-Spreen, I. M. Sokolov, and M. Starnini, "Modeling Echo Chambers and Polarization Dynamics in Social Networks," *Phys Rev Lett*, vol. 124, no. 4, 2020, doi: 10.1103/PhysRevLett.124.048301.

82. Ł. G. Gajewski, J. Sienkiewicz, and J. A. Hołyst, "Transitions between polarization and radicalization in a temporal bilayer echo-chamber model," *Phys Rev E*, vol. 105, no. 2, 2022, doi: 10.1103/PhysRevE.105.024125.

83. T. D. Pilditch, "Opinion Cascades and Echo-Chambers in Online Networks: A Proof of Concept Agent-Based Model," in *CogSci 2017 - Proceedings of the 39th Annual Meeting of the Cognitive Science Society: Computational Foundations of Cognition*, 2017.

84. D. Geschke, J. Lorenz, and P. Holtz, "The triple-filter bubble: Using agent-based modelling to test a meta-theoretical framework for the emergence of filter bubbles and echo chambers," *British Journal of Social Psychology*, vol. 58, no. 1, 2019, doi: 10.1111/bjso.12286.

85. M. Al Atiqi, S. Chang, and H. Deguchi, "Agent-based approach to resolve the conflicting observations of online echo chamber," in *2020 Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems, SCIS-ISIS 2020*, 2020. doi: 10.1109/SCISISIS50064.2020.9322696.

86. J. P. Fränken and T. D. Pilditch, "Cascades across networks are sufficient for the formation of echo chambers: An agent-based model," *JASSS*, vol. 24, no. 3, 2021, doi: 10.18564/jasss.4566.

87. E. E. Santos *et al.*, "Incorporating social theories in computational behavioral models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, pp. 341–349. doi: 10.1007/978-3-319-05579-4_42.

88. E. E. Santos *et al.*, “Modeling Social Resilience in Communities,” *IEEE Trans Comput Soc Syst*, vol. 5, no. 1, pp. 186–199, 2018, doi: 10.1109/TCSS.2017.2780125.

89. E. E. Santos *et al.*, “Modeling insider threat types in cyber organizations,” in *2017 IEEE International Symposium on Technologies for Homeland Security, HST 2017*, 2017. doi: 10.1109/THS.2017.7943445.

90. E. E. Santos *et al.*, “Modeling Emergent Border-Crossing Behaviors during Pandemics,” in *SPIE Defense, Security and Sensing*, 2013.

91. E. E. Santos, E. Santos, Jr., L. Pan, J. T. Wilkinson, J. E. Thompson, and J. Korah, “Infusing Social Networks with Culture,” *IEEE Trans Syst Man Cybern Syst*, vol. 44, no. 1, pp. 1–17, 2014, doi: 10.1109/TSMC.2013.2238922.

92. E. E. Santos *et al.*, “Intent-Driven Behavioral Modeling during Cross-Border Epidemics,” *Proceedings of the 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing*, vol. 11, no. 2, pp. 748–755, 2011, doi: 10.1109/PASSAT/SocialCom.2011.187.

93. W. Goffman and V. A. Newill, “Generalization of epidemic theory: An application to the transmission of ideas,” *Nature*, vol. 204, no. 4955, 1964, doi: 10.1038/204225a0.

94. P. A. Dreyer and F. S. Roberts, “Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion,” *Discrete Appl Math (1979)*, vol. 157, no. 7, 2009, doi: 10.1016/j.dam.2008.09.012.

95. K. M. A. Kabir, K. Kuga, and J. Tanimoto, “Analysis of SIR epidemic model with information spreading of awareness,” *Chaos Solitons Fractals*, vol. 119, 2019, doi: 10.1016/j.chaos.2018.12.017.

96. Y. Liu, B. Wang, B. Wu, S. Shang, Y. Zhang, and C. Shi, “Characterizing super-spreading in microblog: An epidemic-based information propagation model,” *Physica A: Statistical Mechanics and its Applications*, vol. 463, 2016, doi: 10.1016/j.physa.2016.07.022.

97. E. E. Santos *et al.*, “Modeling Complex Social Scenarios Using Culturally Infused Social Networks,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Anchorage, AK, 2011, pp. 3009–3016.

98. E. Santos, Jr. and E. S. Santos, “A Framework for Building Knowledge-Bases Under Uncertainty,” *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 11, no. 2, pp. 265–286, 1999.

99. J. Andreoni and J. Miller, “Giving according to GARP: An experimental test of the consistency of preferences for altruism,” *Econometrica*, vol. 70, no. 2, pp. 737–753, 2002.

100. R. Forsythe, J. Horowitz, N. Savin, and M. Sefton, “Fairness in simple bargaining experiments,” *Games Econ Behav*, vol. 6, pp. 347–369, 1994, doi: 10.1006/game.1994.1021.

101. O. Al-Ubaydli, G. Jones, and J. Weel, “Average player traits as predictors of cooperation in a repeated prisoner’s dilemma,” *Journal of Behavioral and Experimental Economics* , vol. 64, pp. 50–60, 2016, doi: 10.1016/j.socec.2015.10.005.

102. J. M. Digman, “Personality structure: emergence of the five-factor model,” *Annu Rev Psychol*, vol. 41, no. 1, 1990, doi: 10.1146/annurev.ps.41.020190.002221.

103. J. C. Raven, “Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive.,” *MSc Thesis, University of London*, 1936.

104. A. W. Tucker, “The Mathematics of Tucker: A Sampler,” *The Two-Year College Mathematics Journal*, vol. 14, no. 3, 1983, doi: 10.2307/3027092.

105. M. Akın, O. Amil, and M. Özdevecioğlu, “Is Your Manager a Psychopath? An Evaluation of the Relationship between the Personality Types of Managers and Workers and the Levels of Psychopathy,” *Procedia Soc Behav Sci*, vol. 221, 2016, doi: 10.1016/j.sbspro.2016.05.092.

106. P. Babiak, C. S. Neumann, and R. D. Hare, “Corporate psychopathy: Talking the walk,” *Behavioral Sciences and the Law*, vol. 28, no. 2, 2010, doi: 10.1002/bsl.925.

107. C. R. Boddy, “Psychopathy screening for public leadership,” *International Journal of Public Leadership*, vol. 12, no. 4, 2016, doi: 10.1108/ijpl-08-2015-0023.

108. H. S. Cheang and S. H. Appelbaum, “Corporate psychopathy: Deviant workplace behaviour and toxic leaders (part two),” *Industrial and Commercial Training*, vol. 47, no. 5, 2015, doi: 10.1108/ICT-12-2013-0087.

109. A. Cohen, “Are they among us? A conceptual framework of the relationship between the dark triad personality and counterproductive work behaviors (CWBs),” *Human Resource Management Review*, vol. 26, no. 1, 2016, doi: 10.1016/j.hrmr.2015.07.003.

110. A. Gudmundsson and G. Southey, “Leadership and the rise of the corporate psychopath: What can business schools do about the ‘snakes inside’?,” *Journal of*

Social & Behavioural Research in Business Gudmundsson & Southey –Journal of Social & Behavioural Research in Business, vol. 22, no. 2, 2011.

111. C. Mathieu, C. S. Neumann, R. D. Hare, and P. Babiak, "A dark side of leadership: Corporate psychopathy and its influence on employee well-being and job satisfaction," *Pers Individ Dif*, vol. 59, 2014, doi: 10.1016/j.paid.2013.11.010.

112. R. J. Pech and B. W. Slade, "Organisational sociopaths: rarely challenged, often promoted. Why?," *Society and Business Review*, vol. 2, no. 3, 2007, doi: 10.1108/17465680710825451.

113. S. F. Smith and S. O. Lilienfeld, "Psychopathy in the workplace: The knowns and unknowns," *Aggression and Violent Behavior*, vol. 18, no. 2. 2013. doi: 10.1016/j.avb.2012.11.007.

114. A. Wojtczuk-Turek and D. Turek, "Executive Psychopaths. Abusive Behaviour of the Management," *Kwartalnik Ekonomistów i Menedżerów*, vol. 22, no. 4, 2011, doi: 10.5604/01.3001.0009.5538.

115. "Psychopath Definition & Meaning - Merriam-Webster." Accessed: Jun. 18, 2023. [Online]. Available: https://www.merriam-webster.com/dictionary/psychopath

116. L. Wang, S. Q. Ye, K. H. Cheong, W. Bao, and N. gang Xie, "The role of emotions in spatial prisoner's dilemma game with voluntary participation," *Physica A: Statistical Mechanics and its Applications*, vol. 490, 2018, doi: 10.1016/j.physa.2017.08.033.

117. M. H. Ribeiro, P. H. Calais, V. A. F. Almeida, and W. Meira Jr, "' Everything I Disagree With is# FakeNews': Correlating Political Polarization and Spread of Misinformation," *arXiv preprint arXiv:1706.05924*, 2017.

118. A. K. Cybenko and G. Cybenko, "AI and Fake News," *IEEE Intell Syst*, vol. 33, no. 5, 2018, doi: 10.1109/MIS.2018.2877280.

119. K. Shu, S. Wang, and H. Liu, "Exploiting tri-relationship for fake news detection," *arXiv preprint arXiv:1712.07709*, 2017.

120. Q. Zhao, E. Santos, H. Nguyen, and A. Mohamed, "What makes a good summary?," in *Computational Methods for Counterterrorism*, Springer, 2009, pp. 33–50.

121. E. Santos Jr. and H. Nguyen, "Modeling users for adaptive information retrieval by capturing user intent," in *Collaborative and social information retrieval and access: Techniques for improved user modeling*, IGI Global, 2009, pp. 88–118.

122. R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, “Parsing with Compositional Vector Grammars,” in *Proceedings of ACL*, 2013. doi: 10.1017/CBO9781139058452.

123. S. Tabachnik, “Flat Earthers descend on Denver for second annual conference,” The Denver Post. Accessed: Nov. 25, 2018. [Online]. Available: https://www.denverpost.com/2018/11/15/denver-flat-earth-conference/

124. D. Walton, “Argumentation theory: A very short introduction,” in *Argumentation in Artificial Intelligence*, I. Rahwan and G. R. Simari, Eds., 2009, pp. 1–22. doi: 10.1007/978-0-387-98197-0_1.

125. Plato, *Republic, Volume I: Books 1-5*. Cambridge, MA: Loeb Classical Library 237. Cambridge, MA: Harvard University Press, 2013.

126. F. Nietzsche, “On truth and lies in a nonmoral sense,” *Philosophy and Truth. Selections from Nietzsche’s Notebooks of the Early 1870s*, 1873.

127. J. K. Burgoon, J. P. Blair, T. Qin, and J. F. Nunamaker, “Detecting deception through linguistic analysis,” in *International Conference on Intelligence and Security Informatics*, Springer, 2003, pp. 91–101.

128. X. Liu, J. Hancock, G. Zhang, R. Xu, D. Markowitz, and N. Bazarova, “Exploring Linguistic Features for Deception Detection in Unstructured Text,” in *Proceedings of the Rapid Screening Technologies, Deception Detection and Credibility Assessment Symposium*, 2012. doi: 10.1109/ HICSS.2003.1173793.

129. S. Feng, R. Banerjee, and Y. Choi, “Syntactic Stylometry for Deception Detection,” *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics*, 2012, doi: 10.1016/j.rser.2015.12.114.

130. J. T. Hancock, L. E. Curry, S. Goorha, and M. T. Woodworth, “Lies in conversation: An examination of deception using automated linguistic analysis,” in *Annual Conference of the Cognitive Science Society*, 2004. doi: 10.1.1.87.9371.

131. J. T. Hancock, L. E. Curry, S. Goorha, and M. Woodworth, “On lying and being lied to: A linguistic analysis of deception in computer-mediated communication,” *Discourse Process*, 2008, doi: 10.1080/01638530701739181.

132. P. S. Keila and D. B. Skillicorn, “Detecting Unusual and Deceptive Communication in Email,” in *Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research*, 2005.

133. M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards, “Lying words: Predicting deception from linguistic styles,” *Personality and Social Psychology Bulletin*. 2003. doi: 10.1177/0146167203029005010.

134. V. Carofiglio and F. de Rosis, “Exploiting uncertainty and incomplete knowledge in deceptive argumentation,” in *International Conference on Computational Science*, Springer, 2001, pp. 1019–1028.

135. O. Cocarascu and F. Toni, “Detecting deceptive reviews using Argumentation,” in *Proceedings of the 1st International Workshop on AI for Privacy and Security - PrAISe ’16*, 2016. doi: 10.1145/2970030.2970031.

136. D. Li and E. Santos Jr, “Argument formation in the reasoning process: toward a generic model of deception detection,” in *Proceedings of the Workshop on Computational Approaches to Deception Detection*, Association for Computational Linguistics, 2012, pp. 63–71.

137. D. Li, *Deception Detection using Human Reasoning*. Dartmouth College, 2013.

138. T. Groeling, “Media Bias by the Numbers: Challenges and Opportunities in the Empirical Study of Partisan News,” *Annual Reviews of Political Science*, vol. 16, pp. 129–151, 2013, doi: 10.1146/annurev-polisci-040811-115123.

139. V. Vapnik, “The support vector method of function estimation,” in *Nonlinear Modeling*, Springer Science & Business Media, 1998, pp. 55–85.

140. T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1998. doi: 10.1007/s13928716.

141. S. Tong and D. Koller, “Support Vector Machine Active Learning with Applications to Text Classification,” *CrossRef Listing of Deleted DOIs*, 2000, doi: 10.1162/153244302760185243.

142. M. Ott, C. Cardie, and J. T. Hancock, “Negative Deceptive Opinion Spam,” in *Proceedings of NAACL-HLT*, 2013. doi: 10.1016/j.scitotenv.2014.07.054.

143. M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, “Finding deceptive opinion spam by any stretch of the imagination,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, 2011, pp. 309–319.

144. D. Walton, “Deceptive arguments containing persuasive language and persuasive definitions,” *Argumentation*, 2005, doi: 10.1007/s10503-005-2312-y.

145. T. Brader, “Striking a responsive chord: How political ads motivate and persuade voters by appealing to emotions,” *American Journal of Political Science*. 2005. doi: 10.1111/j.0092-5853.2005.00130.x.

146. D. Maillat and S. Oswald, “Defining Manipulative Discourse: The Pragmatics of Cognitive Illusions,” *International Review of Pragmatics*, 2009, doi: 10.1163/187730909X12535267111651.

147. S. Volkova and J. Y. Jang, “Misleading or Falsification? Inferring Deceptive Strategies and Types in Online News and Social Media,” *WWW-2018*, 2018, doi: 10.1145/3184558.3188728.

148. Y. R. Tausczik and J. W. Pennebaker, “The psychological meaning of words: LIWC and computerized text analysis methods,” *J Lang Soc Psychol*, vol. 29, no. 1, pp. 24–54, 2010.

149. J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, “The development and psychometric properties of LIWC2015,” Austin, TX: University of Texas at Austin, 2015.

150. A. Lauscher, H. Wachsmuth, I. Gurevych, and G. Glavaš, “Scientia Potentia Est—On the Role of Knowledge in Computational Argumentation,” *Trans Assoc Comput Linguist*, vol. 10, 2022, doi: 10.1162/tacl_a_00525.

151. G. Zhang, P. Nulty, and D. Lillis, “Enhancing Legal Argument Mining with Domain Pre-training and Neural Networks,” *Journal of Data Mining & Digital Humanities*, vol. NLP4DH, 2022, doi: 10.46298/jdmdh.9147.

152. I. Habernal *et al.*, “Mining Legal Arguments in Court Decisions,” Aug. 2022, Accessed: Jun. 24, 2023. [Online]. Available: https://arxiv.org/abs/2208.06178v2

153. H. Hüning, L. Mechtenberg, and S. Wang, “Detecting Arguments and Their Positions in Experimental Communication Data,” *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.4052402.

154. D. de V. Feijo and V. P. Moreira, “Improving abstractive summarization of legal rulings through textual entailment,” *Artif Intell Law (Dordr)*, vol. 31, no. 1, 2023, doi: 10.1007/s10506-021-09305-4.

155. M. Elaraby and D. Litman, “ArgLegalSumm: Improving Abstractive Summarization of Legal Documents with Argument Mining,” Sep. 2022, Accessed: Jun. 24, 2023. [Online]. Available: https://arxiv.org/abs/2209.01650v2

156. H. Xu and K. Ashley, “Multi-granularity Argument Mining in Legal Texts,” Oct. 2022, Accessed: Jun. 24, 2023. [Online]. Available: https://arxiv.org/abs/2210.09472v2

157. D. A. Broniatowski, J. R. Simons, J. Gu, A. M. Jamison, and L. C. Abroms, “The efficacy of Facebook’s vaccine misinformation policies and architecture during the COVID-19 pandemic,” *Sci Adv*, vol. 9, no. 37, p. eadh2132, Oct. 2023, doi: 10.1126/sciadv.adh2132.

158. M. Cinelli, G. D. F. Morales, A. Galeazzi, W. Quattrociocchi, and M. Starnini, "Echo chambers on social media: A comparative analysis," *arXiv preprint arXiv:2004.09603*, 2020.

159. G. Eady, J. Nagler, A. Guess, J. Zilinsky, and J. A. Tucker, "How many people live in political bubbles on social media? Evidence from linked survey and Twitter data," *Sage Open*, vol. 9, no. 1, p. 2158244019832705, 2019.

160. R. S. Nickerson, "Confirmation Bias: A Ubiquitous Phenomenon in Many Guises," *Review of General Psychology*, vol. 2, no. 2, pp. 175–220, 1998, doi: 10.1037/1089-2680.2.2.175.

161. D. C. Giancoli, *Physics for scientists and engineers with modern physics*, vol. 2. Pearson Education, 2008.

162. N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

163. M. Workman, "An Empirical Study of Social Media Exchanges about a Controversial Topic: Confirmation Bias and Participant Characteristics," 2018.

164. C. R. Sunstein, "The Law of Group Polarization," *SSRN Electronic Journal*, Jul. 2005, doi: 10.2139/ssrn.199668.

165. K. Strandberg, S. Himmelroos, and K. Grönlund, "Do discussions in like-minded groups necessarily lead to more extreme opinions? Deliberative democracy and group polarization," *International Political Science Review*, vol. 40, no. 1, pp. 41–57, 2019, doi: 10.1177/0192512117692136.

166. S. Moscovici and M. Zavalloni, "The group as a polarizer of attitudes.," *J Pers Soc Psychol*, vol. 12, no. 2, p. 125, 1969.

167. L. M. van Swol, "Extreme members and group polarization," *Soc Influ*, vol. 4, no. 3, pp. 185–199, 2009, doi: 10.1080/15534510802584368.

168. M. J. Blanca Mena, R. Alarcón Postigo, J. Arnau Gras, R. Bono Cabré, and R. Bendayan, "Non-normal data: Is ANOVA still a valid option?," *Psicothema*, 2017.

169. S. Weerahandi, "ANOVA under Unequal Error Variances," *Biometrics*, vol. 51, no. 2, pp. 589–599, 1995, doi: 10.2307/2532947.

170. M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artif Intell Rev*, vol. 55, no. 7, 2022, doi: 10.1007/s10462-022-10144-1.

171. D. Spohr, “Fake news and ideological polarization: Filter bubbles and selective exposure on social media,” *Business Information Review*, vol. 34, no. 3, 2017, doi: 10.1177/0266382117722446.

172. P. Moravec, R. Minas, and A. R. Dennis, “Fake News on Social Media: People Believe What They Want to Believe When it Makes No Sense at All,” *SSRN Electronic Journal*, 2018, doi: 10.2139/ssrn.3269541.

173. B. Larsen and J. Narayan, “Generative AI – a game-changer society needs to be ready for | World Economic Forum,” DAVOS. Accessed: Jul. 02, 2023. [Online]. Available: https://www.weforum.org/agenda/2023/01/davos23-generative-ai-a-game-changer-industries-and-society-code-developers/

174. Y. Bang *et al.*, “A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity,” Feb. 2023, Accessed: Jun. 18, 2023. [Online]. Available: https://arxiv.org/abs/2302.04023v2

175. G. E. P. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

176. E. Santos, Jr., J. T. Wilkinson, and E. E. Santos, “Bayesian Knowledge Fusion,” in *Proceedings of the 22nd International FLAIRS Conference*, Sanibel Island, FL: AAAI Press, 2009, pp. 559–564.