Identifying and Counteracting Bias in the News

CT5117 Research Project

Caroline Manghan

19231865

Introduction

Bias can infiltrate the news in a variety of ways and can have a stark effect on readers, who rely on the news for important information.This phenomenon can be readily observed within political news articles, where such bias can have important effects on reader opinions and voter behaviours. Thus, it is important such bias be identified and counteracted.

This report introduces the News Aggregator Prototype, a tool designed to achieve this goal. Users can search a term of their choosing, and the aggregator will return to them an unbiased selection of articles containing the term. To measure the prototype's effectiveness a survey of 55 participants was conducted.

Motivation

Despite the rise of social media, Americans still rely heavily on news agencies. Each day, an estimated 75% of Americans tune into newscasts and online news sources. In particular, Americans regularly consume political news content through these sources (American Press Institute, 2014). This content, according to, de Arruda, et. al. (2020), is vital for consumers to form political opinions. Most Americans do not personally meet with political leaders or candidates to learn their platforms and policies, and, as such, must rely on second hand sources for information. (Eberl, et. al. 2016).

However, due to media bias and the explosion of so-called "fake news" Americans may not be presented with the neutral information needed to form unbiased opinions. This is not a new phenomenon. Lippmann warned in 1922 of the dangerous ability of the media to set the political agenda, and Becker & McCombs found the media played a role in shaping voters' views of candidates in the 1976 presidential primaries (Lippmann, 1922, Becker & McCombs, 1978). However, Crawford et. al. (2018) argue these dangers may become more pronounced with the influence of Social Media. Consumers may find themselves in an "echo-chamber" in which news shared by friends, promoted by celebrities the user has "liked", or recommended by sorting algorithms, conforms to their already held beliefs (Crawford et. al., 2018). In a study conducted by the Pew Research Center, researchers divided participants by political affiliation. They found participants were likely to consult information from news media, social media, and word of mouth produced by like minded individuals. Their findings were especially profound in participants with more extreme views. The sources consulted by participants with right leaning ideologies had little overlap with participants on the left (Pew Research Center, 2014).

Additionally, 90% of news corporations are controlled by just six companies. This, according to Hamborg et. al. (2018), increases the likelihood that these news outlets may be intentionally biased. With few challengers, such bias may permeate into public discourse and opinion (Hamborg et. al., 2018).

A public which is unable to form unbiased opinions can have pronounced effects on society. For example, McKeever et. al. (2012) contend that the way in which the media reports on refugees impacts how others see them. An outlet's choice to use pictures of smiling families and terms such as 'Economic Migrant' or pictures of aggressive looking individuals and terms such as 'Illegal Immigrant' may influence how consumers perceive refugees (McKeever et. al., 2012, Hamborg et. al., 2018). In another example, The Washington League for Increased Transparency and Ethics ("WASHLITE") argue the language media outlet Fox News used in discussing COVID-19 was maliciously deceitful and erroneous. The language, they argue in a recently filed lawsuit, misled viewers and resulted in a "serious public health issue" (WASHLITE, 2020). In such cases as these, media bias has profound effects on how society views groups of people or reacts in a health emergency.

As the media can influence public opinion, it can thus influence voting decisions and elections. This phenomenon was criticized heavily in the 2016 election in which an unprecedented influx of media from a variety of sources, including many categorized as "fake news", was presented to consumers and shared across social media platforms (Xu et. al., 2018). The public has taken special interest in the issue, as many worried it affected the ultimate outcome of the election (Hirning et. al., 2017). This influence, according to Hamborg et. al. (2018) challenges the foundations of the U.S. democracy. If voters are unable to form unbiased opinions, they argue, the public opinion expressed at the polls is not reflective of the true public opinion. As such, it is crucial voters receive an unbiased view of the news.

Background

Bias is difficult to detect as it is often subtle and can take many forms. This report focuses on intentional bias, in which news organizations consciously inject bias to achieve a goal. News organizations are motivated by numerous factors to intentionally inject bias into their product. First, they may be motivated to reach a target audience who search out news aligning with their already held views. Alternatively, the news organization may be motivated by advertising funds (Hamborg et. al., 2018). Advertising funds may entice news networks to create sensationalized or misleading headlines in order to generate more views (Chen et. al., 2015, Blom & Hansen, 2014). Advertisers may also discourage organizations from reporting on stories which negatively frame the advertiser. Finally, individuals within the news organization including journalists, producers, etc. may inject bias which aligns with their personally held beliefs (Hamborg, et. al., 2018).

Intentional bias can be further categorized into spin bias and ideology bias. Spin bias occurs when a news organization manipulates a story to make it more memorable. Ideological bias occurs when a news organization manipulates a story for the purpose of promoting an opinion (Hamborg et. al., 2018). This report focuses on the identifying and counteracting the effects of ideology bias.

Finally, ideology bias can be further divided into a number of categories including selection bias, coverage bias, and framing bias. Selection bias, or gatekeeping bias, is introduced when a news organization chooses whether or not to cover topics. Selection bias can be topic related or geographical. For example, consumers close to the site of a natural disaster are likely to be more interested in the government's response than those far away (Lazaridou & Krestel, 2016). Coverage bias is observed through the time and space a story is allotted. For example, news organizations may choose to display an article that aligns with their desired viewpoint more prominently on the front page or for a longer time. Framing bias, or statement bias is concerned with how the content is reported (de Arruda et. al., 2020). This type of bias can occur through the language a news article uses to describe political parties, politicians, and events or through reinforcing certain statements (Trumper, et. al. 2013, Morstatter et. al., 2018).

This report focuses on bias at the ideology level and does not further categorize bias into selection bias, coverage bias, or framing bias.

Purpose

The goal of this report is to identify, categorize, and counteract biased news articles. This is achieved through the News Aggregator Prototype, detailed in the sections below. The aggregator categorizes each article based on its bias. It then presents the user with a variety of articles on the same topic, each from a different political perspective, in order to eliminate the "echo chamber" effect.

Related Work

The topic of bias in the media has been debated since the invention of the printing press in the fifteenth century. In 1671, Governor Sir William Berkely of Virginia accused the press of printing “libels against the best government” (Long, 1918). In a study of all articles printed by the Washington Post and the Washington Times between September 1 and November 10, 1988, Kenney and Simpson (1993) found evidence of selection bias, as key events of the 1988 election were not covered by either paper (Kenney & Simpson, 1993). More recently, researchers have focused on automating bias detection.

Budak et. al. (2016) use supervised learning algorithms and crowdsourcing to identify and examine news organizations overall bias slant and slant by topic. They accomplish this using a corpus of news articles collected from Bing Toolbar users. The authors amassed the URLs of all news articles which had views from at least 10 unique Bing users. From this corpus, they identified politically relevant stories using two binary classifiers, which were trained on article features. Article features were determined by collecting the most frequent words appearing amongst the first 100 words of each article. Labels were crowdsourced using Amazon Mechanical Turk workers who assigned each article a label from a set of categories including news, finance, lifestyle, etc. The first classifier identified news stories, and the second identified which news stories were political. The authors then used crowdsourcing to categorize the articles in the remaining data set by topic within politics, such as human rights, healthcare, etc. Finally, a weighted sample of articles was taken from this dataset to examine for bias.

To identify bias, Budak et. al. (2016) used crowdsourcing to assign each article a partisanship and to determine if it portrayed the Republican or Democratic party negatively. The bias of each article was then aggregated to reflect on the news organization. The authors found the slant of news organizations to align with past research. For example, they found Breitbart News to lean far to the right and Daily Kos to lean far to the left. When broken down by topic, the authors also found little evidence of selective bias. Budak et. al. (2016) contend using crowdsourcing increases the scalability compared to using undergraduate students (Budak et. al., 2016). However, the authors’ methods were time consuming and required 749 online participants.

In contrast, Pryzant et. al. use less manual methods to determine and neutralize bias injected through language. The authors’ goal is to replace text and headlines that convey an opinion, such as ‘Candidate A exposed as unprincipled,’ to more neutral statements such as

‘Candidate A described as unprincipled.’ To do so, they generated a corpus of 180,000 biased and neutralized sentence pairs. The sentence pairs were accumulated from over 420,000 Wikipedia revisions between 2004 and 2019 (Pryzant et. al., 2020). These revisions were made in order to align Wikipedia articles with Wikipedia’s neutral point of view, which requires content be written “fairly, proportionately, and, as far as possible, without editorial bias” (Pryzant et. at., 2020, Wikipedia, 2020).

Pryzant et. al. (2020) next propose two algorithms, one MODULAR and one CONCURRENT, to debias text. The MODULAR algorithm is two stages. The first stage is BERT-based detection, and the second is LSTM-based editing. The authors found the MODULAR algorithm allowed greater human control and interpretability. In contrast, the CONCURRENT algorithm, which uses a BERT encoder to identify subjectivity, was found to be simple to train and operate. When using these algorithms on external datasets, Pryzant et. al. found the MODULAR algorithm outperformed the CONCURRENT algorithm when reducing bias, but the CONCURRENT algorithm better retained the original text’s meaning (Pryzant et. at., 2020).

Similarly, Morstatter et. al. (2018) use automated methods to detect framing bias. This type of bias, they contend, is difficult to discern, as it is not as easily measured as the length of an article or the number of articles on a single topic. Instead, Morstatter et. al. measure the focus and emphasis of the article.

The authors began by assembling a large corpus on the Ballistic Missile Defense System in Europe composed of articles originating from many different countries. Each sentence within an article was hand coded and translated into English. Each sentence was tagged with a frame from a predetermined list of frames, such as General Threat, Political Tensions, etc., as well as a positive or negative polarity. This model is then used to determine in a random 10% of sentences within the corpus if a sentence has a frame, the frame’s type, and the frame’s polarity, using machine learning. When applied to articles outside of the corpus, the model was able to successfully analyze the articles’ framing, albeit with a slight drop in performance (Morstatter et. al., 2018).

This report builds on related work in identifying bias and also proposes a method of decreasing the effects of bias through the bias aware News Aggregator Prototype.

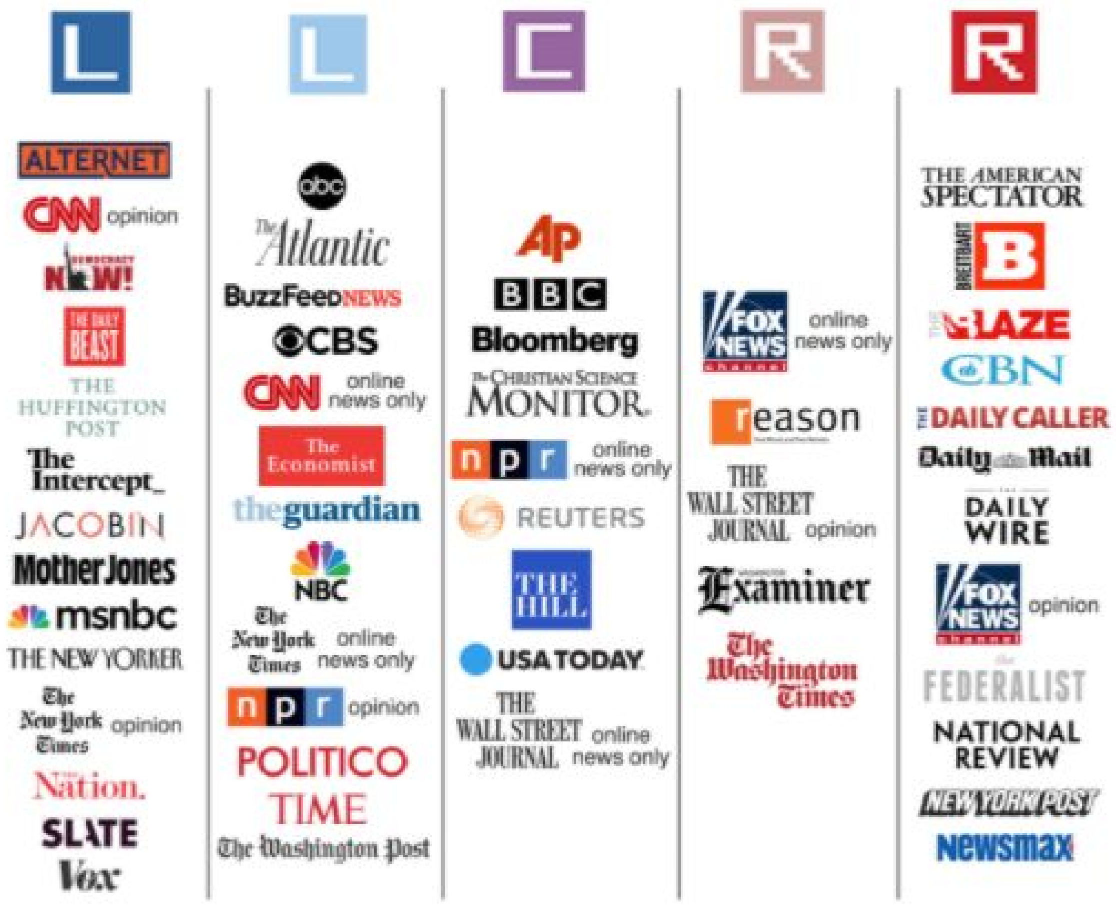
Methodology

To categorize news organizations, this report uses classifications produced by AllSides and displayed in the AllSides Media Bias Chart. AllSides determines bias through a number of methods. First, they use crowdsourcing through their Blind Bias Survey. This survey prompts readers across the political spectrum to read and rate articles without a listed source. The resulting data is then normalized to prevent nefarious ratings. Additionally, AllSides combines data from third parties including academic research, surveys, and analysis. Further, AllSides uses ratings from their own editorial staff, whose opinions range across the political spectrum, and community feedback, which warns if a rating may be incorrect. Finally, AllSides staff conducts independent research and analysis to determine bias (AllSides, 2020).

AllSides contends it is “more comprehensive in its methodology than any other media bias chart on the Web” (AllSides, 2020) and their results have been used in similar studies including Chen, et. al. (2018) and Risius, et. al. (2018). Further, studies by Ribeiro et. al. (2018) and Bentley et. al. (2019) independently examine bias in news organizations and, when compared with AllSides, they found Pearson correlations of .77 and .81 respectively (Ribeiro et. al., 2018, Bentley et. al., 2018). Given these strong correlations, AllSides’ positive reputation in the academic community, and limitations including time and funding which prevent the development of a unique dataset, this report uses AllSides’ findings in the development of the News Aggregator Prototype.

AllSides classifies news organizations into five categories, which are referred to in this report as: Far Left, Left, Center, Right, and Far Right. The classifications are detailed in figure 1 below.

Figure 1



Source: (AllSides, 2020)

To develop the News Aggregator Prototype I first developed a corpus of news articles from the above news organizations. To limit selection bias, I chose the first 10 articles to appear on each organization's website when filtered to the 'Politics' or 'US Politics' section. The articles were collected from between March 11, 2020 and March 23, 2020 and amalgamated to a single database.

Each article within the database is labeled with seven characteristics: Database ID, Title, Author, Source, Weight, URL, and Content. Articles were assigned a Database ID in the order they were collected. This ID is an arbitrary number used to conveniently identify articles and is not a measure of bias. Weights were assigned to articles based on the source's position in Figure 1. Articles from sources in the Far Left category were assigned a weight of -2, articles from sources in the Left category were assigned a weight of -1, articles from sources in the

Center category were assigned a weight of 0, articles from the Right category were assigned a weight of 1, and articles from the Far Right category were assigned a weight of 2. The numbers were chosen based on the linear depiction of sources in Figure 1. Whether an article is assigned a positive or negative number is not indicative of the article's accuracy or any personal views on the source.

The News Aggregator Prototype, which pulls from this database of articles, begins by asking the user for a term to search within the database. The user can search any term based on their topic preferences. The user is then presented with two options:

1. Enter 1 to search for the requested term in the title

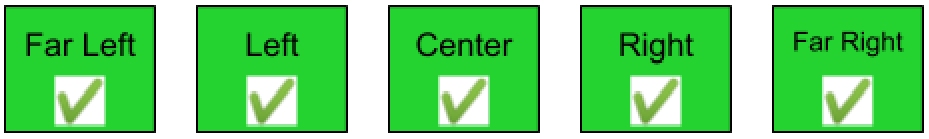
By selecting this option the News Aggregator will search for the entered term within each article stored in the database. The program will collect all results, even articles with headlines in which the search term is embedded within a larger word. For example, if the user searches the term 'Corona,' headlines with the term 'Coronavirus' will be collected, as it contains the requested inner term.

2. Enter 2 to search for the requested term in the article body

If the user selects this option, the News Aggregator will search for the entered term within the entire article body. Similar to the first option, the program will collect all articles that contain the term on its own or the term embedded within a larger word or phrase.

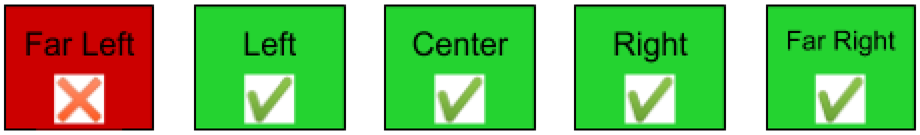
Once collected, the articles containing the search term are grouped by their weight and presented to the user so the total sum of weights is zero. In order to prevent information overload, which, according to Holton & Chyi (2012), "occurs when the amount of available content becomes difficult for an individual to process, often causing negative feelings on the end of the customer," the user is presented with a maximum of five articles at a time. Depending on the chosen search term, one of the following scenarios will occur:

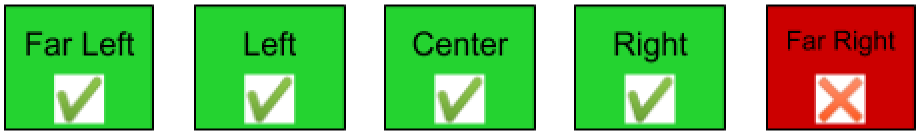
a) A search term results in at least one matching article from each political affiliation.

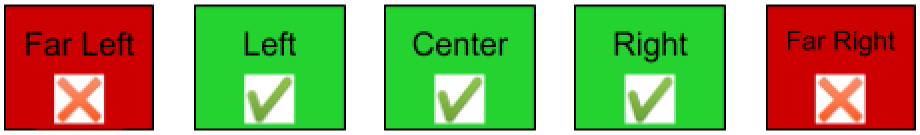
i) 

In this scenario, the user will be presented with five articles, one from each affiliation.

b) A search term does not result in at least one matching article in the Far Left and/or Far Right categories.

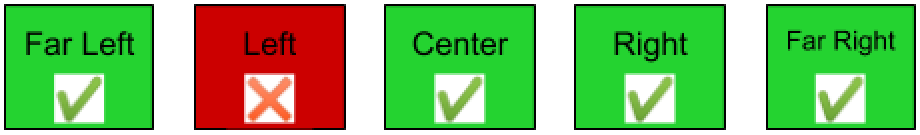
i) 

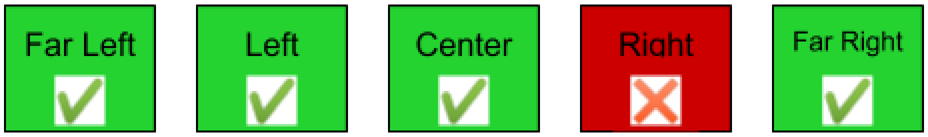
ii) 

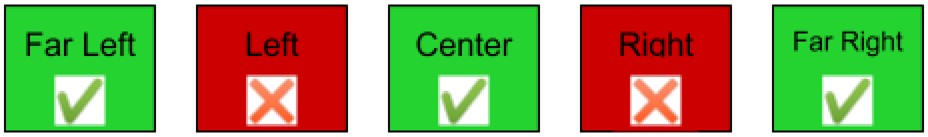
iii) 

In each of these three scenarios, the user will be presented with three articles, one from each the Left, Center, and Right.

c) A search term does not result in at least one matching article in the Left and/or Right categories.

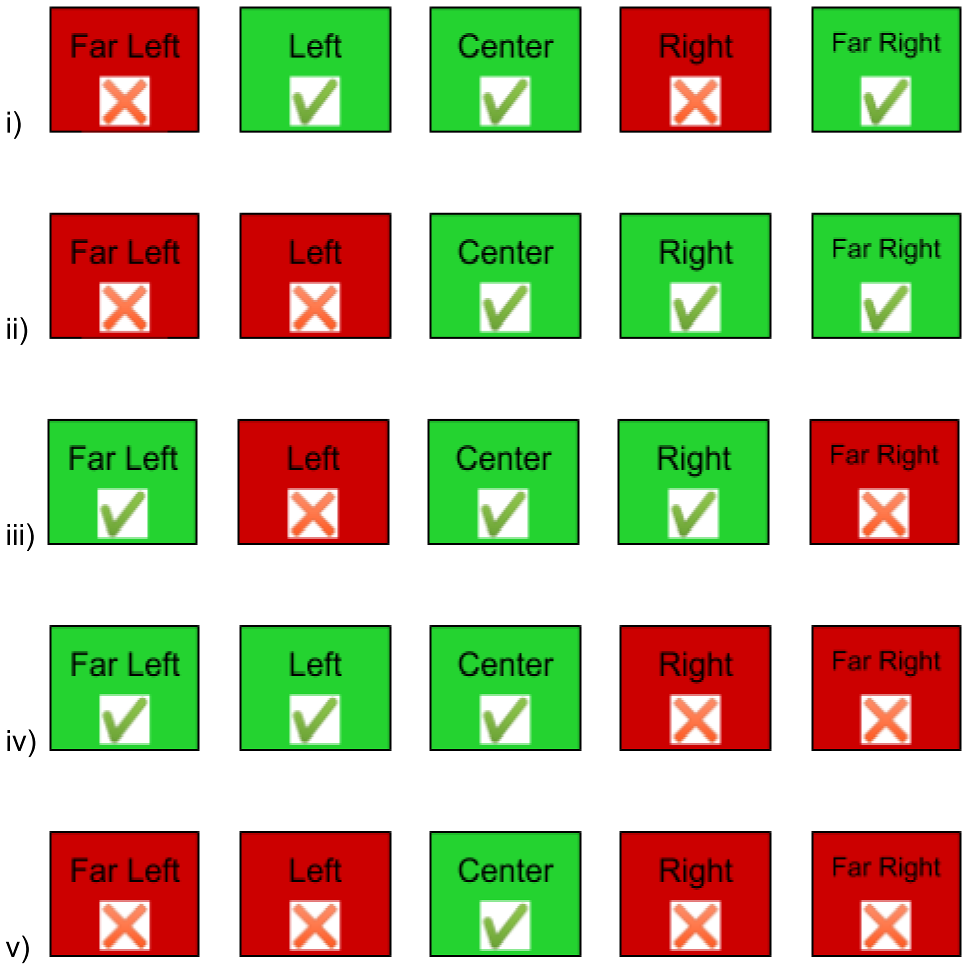
i) 

ii) 

iii) 

In each of these three scenarios, the user will be presented with three articles, one from each the Far Left, Center, and Far Right.

d) A search term does not result in at least one matching article in the Left and/or Right categories, and the search term does not result in at least one matching article in the Far Left and/or Far Right categories.


i) Far Left ✗ Left ✓ Center ✓ Right ✗ Far Right ✓
ii) Far Left ✗ Left ✗ Center ✓ Right ✓ Far Right ✓
iii) Far Left ✓ Left ✗ Center ✓ Right ✓ Far Right ✗
iv) Far Left ✓ Left ✓ Center ✓ Right ✗ Far Right ✗
v) Far Left ✗ Left ✗ Center ✓ Right ✗ Far Right ✗

In any of these five scenarios, the user will only be presented with a single article from the center category.

In each of these scenarios, if a term matches with more than one article within a single affiliation, an article will be chosen at random from all matching articles to be displayed.

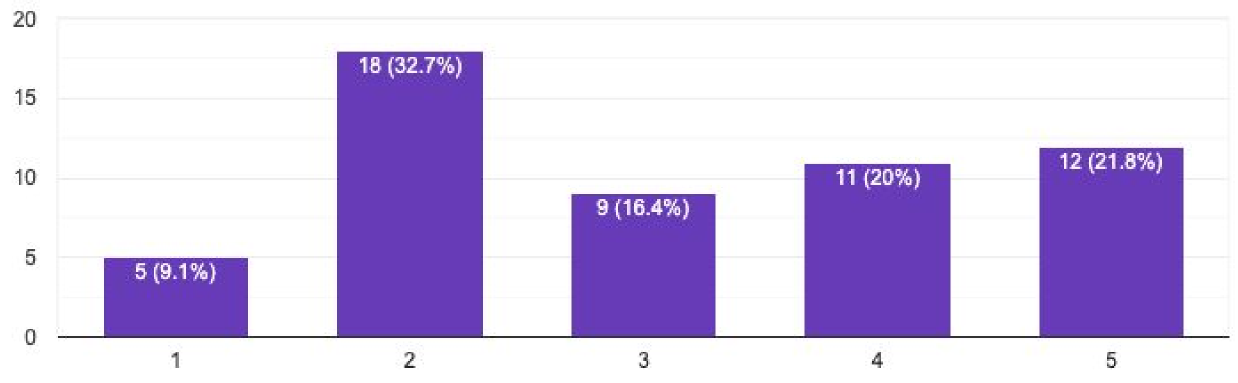
Alternatively a search term does not result in any matches, the user is notified and the program terminates. The program also terminates in any situations where results do not include a result in the center category. This is a design choice intended to ensure a user is not only presented with extreme positions. However, this could be modified in future versions.

Results

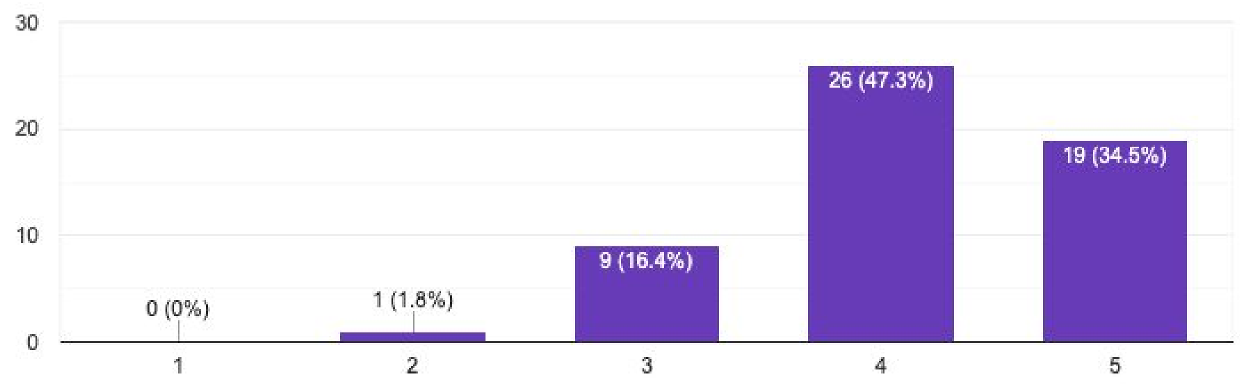
In order to measure the results of the News Aggregator Prototype, I conducted a survey of 55 participants. Participants were selected on a volunteer basis, and each participant was asked to use the prototype and respond to questions regarding their news consumption behaviour with and without the News Aggregator Prototype. Participants were asked only four questions in an attempt to elicit high participation rates. The questions are as follows:

1) Without the News Aggregator, how likely are you to consult a variety of news sites on a single topic?
2) Does using the News Aggregator make you more aware of potential bias in an article?
3) If you chose to reveal the article's sources, did the News Aggregator prompt you to read an article from a source you do not normally consult?
4) Using the News Aggregator, how likely were you to consult more than one of the articles provided?

In Question 1, participants were asked to rate, on a scale of 1 to 5, how likely they are to consult multiple sources on a topic without a bias aware aggregator. 1 represented not very likely, and 5 represented very likely. Responses were varied, with 41.8% of participants responding that they are unlikely to consult multiple sources (responding with a 1 or 2) and 41.8% reporting they are likely to consult multiple sources (responding with a 4 or 5).

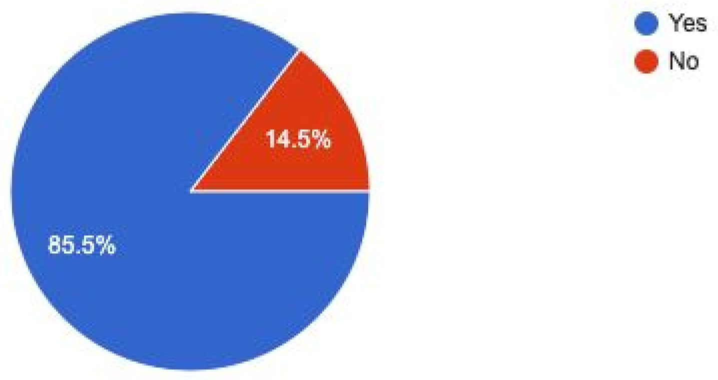


However, when using the News Aggregator, as measured in Question 4, respondents were much more likely to consult multiple sources. Here, 81.8% reported they were likely to consult multiple sources, with only 1.8% of participants reporting the opposite.



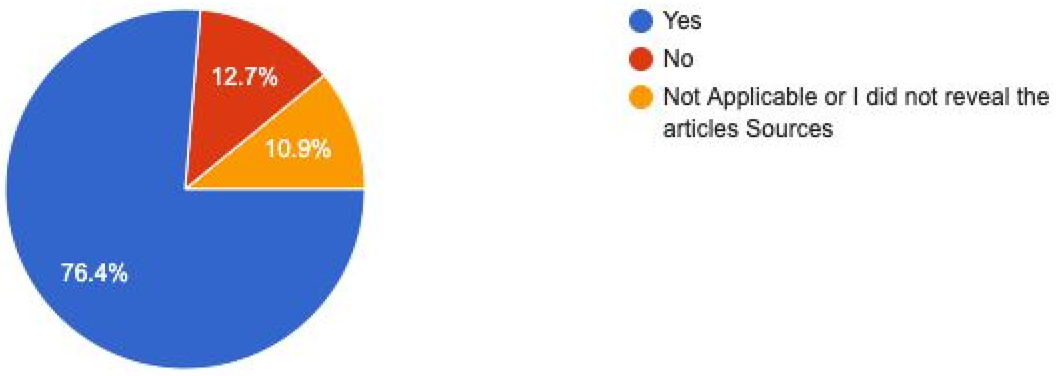
This significant increase illustrates how tools such as the News Aggregator can be used to encourage readers to consume a wider variety of media instead of relying on a single source.

Additionally, 85.5% of survey participants reported the News Aggregator Prototype made them more aware of potential bias in an article.



This result shows the act of simply using a technology such as this can lead to more bias aware consumers.

Finally, 85.7% of participants who chose to reveal the source of an article reported consulting a source through the News Aggregator Prototype that they would not normally consult.



This illustrates how the News Aggregator Prototype can successfully avoid an “echo chamber” effect, in which consumers only consult media which aligns with their already held beliefs.

<u>Further Research</u>

The above results were calculated from a sample of 55 volunteer users. For future studies I would recommend a larger sample chosen at random to increase confidence in the results. Future surveys could also request demographic information to further categorize results by features such as age and gender.

Additionally, although the News Aggregator Prototype shows promising preliminary results, more research is needed to fully gauge its effects and usefulness. First, research is needed around consumers’ willingness to use a tool like the News Aggregator Prototype. Although the prototype has been shown to encourage users to consult a wide variety of sources, users may not be willing to use the technology. Other, non-bias aware, aggregators such as Yahoo News and Google News may be more convenient and user friendly. Additionally, users may not want to read articles promoting a viewpoint which contradicts their own. For example, a user with a far left political affiliation may not be willing to use a tool which is likely to produce articles from far right sources such as Breitbart News Network or The Daily Mail. Thus, further research should compare the News Aggregator Prototype to other currently available aggregators.

Finally, I recommend additional research regarding the effects of using the News Aggregator Prototype on public opinion and voter behaviour. The survey results show the aggregator is successful in encouraging users to consult more varied sources than they normally would. However, additional research would be required to demonstrate if this result translates to changed behaviour.

Conclusion

Bias in news media often goes undetected and can permeate public opinion. As such, it can affect voter behaviour and decisions. To counteract such effects, this report produced the New Aggregator Prototype, a bias aware aggregator designed to detect bias and generate an unbiased selection of articles for readers. Bias was detected based on an article’s source, which was weighted by its political affiliation. The unbiased selection ensured readers were always presented with a neutral weighting.

The News Aggregator Prototype was found to make users more aware of potential bias. Additionally, the tool prompted users to read a wider variety of sources, including sources they do not normally consult.

References

(WASHLITE) WASHINGTON LEAGUE FOR INCREASED TRANSPARENCY AND ETHICS v FOX NEWS, FOX NEWS GROUP, FOX NEWS CORPORATION, RUPERT MURDOCH, AT&T TV, COMCAST. 2020, COMPLAINT FOR DECLARATORY RELIEF AND VIOLATION OF THE WASHINGTON STATE CONSUMER PROTECTION ACT (RCW 19.86). THE SUPERIOR COURT OF WASHINGTON COUNTY OF KING.

Allsides Media Bias Ratings". 2020. Allsides. https://www.allsides.com/media-bias/media-bias-ratings.

BECKER, LEE B., and MAXWELL E. McCOMBS. 1978. "THE ROLE OF THE PRESS IN DETERMINING VOTER REACTIONS TO PRESIDENTIAL PRIMARIES". *Human Communication Research* 4 (4): 301-307. doi:10.1111/j.1468-2958.1978.tb00716.x.

Bentley, Frank, Katie Quehl, Jordan Wirfs-Brock, and Melissa Bica. 2019. "Understanding Online News Behaviors". In *HI Conference On Human Factors In Computing Systems Proceedings (CHI 2019),*. G.

Blom, Jonas Nygaard, and Kenneth Reinecke Hansen. 2014. "Click Bait: Forward-Reference As Lure In Online News Headlines."

Budak, Ceren, Sharad Goel, and Justin M. Rao. 2016. "Fair And Balanced? Quantifying Media Bias Through Crowdsourced Content Analysis". *Public Opinion Quarterly* 80 (S1): 250-271. doi:10.1093/poq/nfw007.

Chen, Wei-Fan, Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. 2018. In *The 11Th International Natural Language Generation Conference*.

Chen, Yimin, Niall J. Conroy, and Victoria L. Rubin. 2020. "Misleading Online Content: Recognizing Clickbait As “False News”."

Crawford, Gregory S., and Vardges Levonyan. 2018. "Media Bias In Public Service Broadcasting: Evidence From The BBC."

de Arruda, Gabriel Domingos, Norton Trevisan Roman, and Ana Maria Monteiro. 2020. "Analysing Bias In Political News". *Journal Of Universal Computer Science,* 26 (2): 173-199.

Eberl, Jakob-Moritz, Markus Wagner, and Hajo G. Boomgaarden. 2016. "Are Perceptions Of Candidate Traits Shaped By The Media? The Effects Of Three Types Of Media Bias". *The International Journal Of Press/Politics* 22 (1): 111-132. doi:10.1177/1940161216674651.

Hamborg, Felix, Karsten Donnay, and Bela Gipp. 2018. "Automated Identification Of Media Bias In News Articles: An Interdisciplinary Literature Review". *International Journal On Digital Libraries* 20 (4): 391-415. doi:10.1007/s00799-018-0261-y.

Hirning, Nicholas P., Andy Chen, and Shreya Shankar. 2017. "Detecting And Identifying Bias-Heavy Sentences In News Articles."

Holton, Avery E., and Hsiang Iris Chyi. 2012. "News And The Overloaded Consumer: Factors Influencing Information Overload Among News Consumers". Cyberpsychology, Behavior, And Social Networking 15 (11): 619-624. doi:10.1089/cyber.2011.0610.

"How Allsides Rates Media Bias: Our Methods". 2020. *AllSides*. https://www.allsides.com/media-bias/media-bias-rating-methods.

Kenney, Keith, and Chris Simpson. 1993. "Was Coverage Of The 1988 Presidential Race By Washington's Two Major Dailies Biased?". *Journalism Quarterly* 70 (2): 345-355.

Lazaridou, Konstantina, and Ralf Krestel. 2016. "Identifying Political Bias In News Articles."

Lippmann, Walter. 1922. *Public Opinion*. New York: Harcourt, Brace & Co.

Long, Jos. R. 1918. "Freedom Of The Press". *Virginia Law Review* 4: 225-246.

McKeever, Brooke Weberling, Daniel Riffe, and Francesca Dillman Carpentier. 2012. "Perceived Hostile Media Bias, Presumed Media Influence, And Opinions About Immigrants And Immigration". *Southern Communication Journal* 77 (5): 420-437. doi:10.1080/1041794x.2012.691602.

Morstatter, Fred, Liang Wu, Uraz Yavanoglu, Stephen R. Corman, and Huan Liu. 2018. "Identifying Framing Bias In Online News". *ACM Transactions On Social Computing* 1 (2): 1-18. doi:10.1145/3204948.

"Neutral Point Of View". 2020. *En.Wikipedia.Org*. https://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view.

Pew Research Center. 2014. "Political Polarization & Media Habits". https://www.pewresearch.org/wp-content/uploads/sites/8/2014/10/Political-Polarization-and-Media-Habits-FINAL-REPORT-11-10-14-2.pdf.

Pryzant, Reig, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. 2020. "Automatically Neutralizing Subjective Bias In Text". In *34Th AAAI Conference On Artificial Intellegence*.

Ribeiro, Filipe N., Lucas Henrique, Fabricio Benevenuto, Abhijnan Chakraborty, Juhi Kulshrestha, Mahmoudreza Babaei, and Krishna P. Gummadi. 2018. "Media Bias Monitor: Quantifying Biases Of Social Media News Outlets At Large-Scale". In *Twelfth International AAAI Conference On Web And Social Media (ICWSM 2018)*.

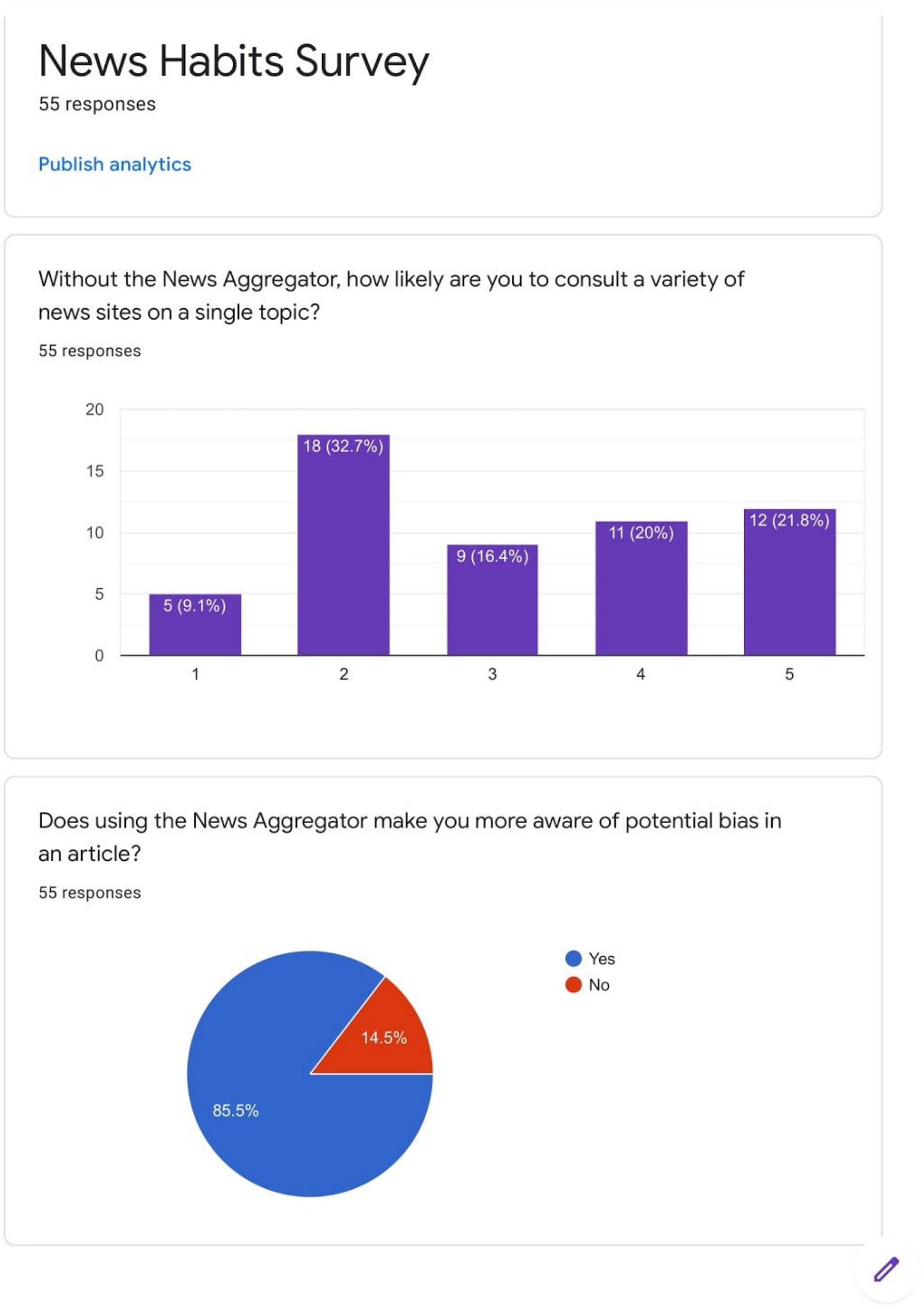
Risius, Marten, Okan Aydinguel, and Maximilian Haug. 2019. "TOWARDS AN UNDERSTANDING OF CONSPIRACY ECHO CHAMBERS ON FACEBOOK". In *27Th European Conference On Information Systems (ECIS)*.

Saez-Trumper, Diego, Carlos Castillo, and Mounia Lalmas. 2013. "Social Media News Communities: Gatekeeping, Coverage, And Statement Bias."

"The Personal News Cycle: How Americans Choose To Get Their News". 2020. *American Press Institute*. https://www.americanpressinstitute.org/publications/reports/survey-research/personal-news-cycle/single-page/.

Xu, Kuai, Feng Wang, Haiyan Wang, and Bo Yang. 2020. "Detecting Fake News Over Online Social Media Via Domain Reputations And Content Understanding". *Tsinghua Science And Technology* 25 (1): 20-27. doi:10.26599/tst.2018.9010139.

Appendix I: News Habits Survey



News Habits Survey

55 responses

Publish analytics

Without the News Aggregator, how likely are you to consult a variety of news sites on a single topic?

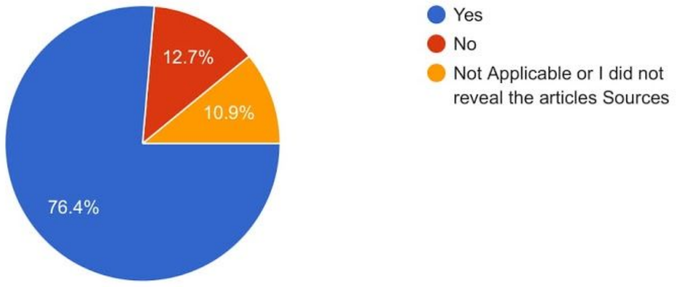
55 responses

Does using the News Aggregator make you more aware of potential bias in an article?

55 responses

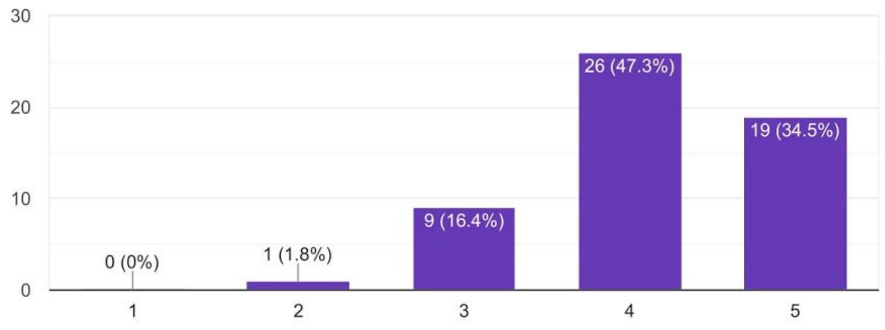
If you chose to reveal the article's sources, did the News Aggregator prompt you to read an article from a source you do not normally consult?

55 responses



Using the News Aggregator, how likely were you to consult more than one of the articles provided?

55 responses



Appendix II: News Aggregator Prototype Code

Main Class

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import java.util.function.Function;
import java.util.stream.Collectors;
/**
 *
 *
 *
 * @author Caroline Manghan
 *
 */
public class Main {

        static List<Article> itemList = Arrays.asList();
        static Article article = new Article();



        public static void main(String[] args) throws IOException {

        //String to hold the neame of the file being read in
        String fileName = "NewsArticlesProjectCSV.csv";

        //call the processInputFile and pass the file's name
        processInputFile(fileName);
        //call the promptUser method
        promptUser();

        }

        //method to input articles
        private static void processInputFile(String fileName) {

        //read in data from the file
        try(BufferedReader br = new BufferedReader(new InputStreamReader(new
FileInputStream(fileName)))) {

        //read in the data, skipping the first line (the first line holds column titles)
        itemList=br.lines().skip(1).map(mapToItem).collect(Collectors.toList());
        br.close(); //close the buffered reader

        } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        }
        }
        //function to map the data to an article object
        private static Function<String, Article> mapToItem = (line) -> {
        //split the csv data by comma
        String [] dataIndex = line.split(",");
        Article item = new Article();

        //set the attributes according to their index in the inputed data
        item.setDatabaseID(dataIndex[0]);
        item.setTitle(dataIndex[1]);
        item.setAuthor(dataIndex[2]);
```

```
	item.setSource(dataIndex[3]);
	item.setWeight(dataIndex[4]);
	item.setURL(dataIndex[5]);
	item.setContent(dataIndex[6]);

	return item;
	};

	//promptUser method interacts with the user
	public static void promptUser() {
	Scanner scanner = new Scanner(System.in);

	//ask the user which term they would like to search and set it to the variable term
	System.out.println("Enter term to search");
	String term = scanner.nextLine();

	//ask the user if they would like to search that term in the title or the article body
	System.out.println("Enter 1 to search for '" + term + "' in the article title. Enter 2 to search in
the article body");
	int where = scanner.nextInt();

	//if the user wants to search the title, call the searchTitle method
	if (where ==1) {
		article.searchTitle(itemList, term);
	}

	//if the user wants to search the body of the article, call the searchBody method
	else if (where == 2) {
		article.searchBody(itemList, term);
	}

	//if the user enters anything else alert them of the invalid input and call the method again
	else {
		System.out.println("Invalid input. Please try again");
```

```
			promptUser();
		}
		}

}
```

Main Class

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.ThreadLocalRandom;

import org.apache.commons.lang3.text.~~WordUtils~~;

public class Article {

		//variables set to private for protection
		private String Title;
		private String Author;
		private String Source;
		private String Weight;
		private String URL;
		private String Content;
		private String DatabaseID;

		//ArrayLists to hold articles when read in and split by political affiliation
		ArrayList <Article> farLeftList = new ArrayList <Article>();
		ArrayList <Article> leftList = new ArrayList <Article>();
		ArrayList <Article> centerList = new ArrayList <Article>();
		ArrayList <Article> rightList = new ArrayList <Article>();
		ArrayList <Article> farRightList = new ArrayList <Article>();
		ArrayList <Article> combinedList = new ArrayList <Article>();
```

```
//opening scanner for interactive capabilities
Scanner scanner = new Scanner (System.in);

//empty default constructor
public Article() {

}

//getter and setter for title
public String getTitle() {
return Title;
}
public void setTitle(String title) {
Title = title;
}

//getter and setter for author
public String getAuthor() {
return Author;
}
public void setAuthor(String author) {
Author = author;
}

//getter and setter for source
public String getSource() {
return Source;
}
public void setSource(String source) {
Source = source;
}
```

```
//getter and setter for weight
public String getWeight() {
return Weight;
}
public void setWeight(String weight) {
Weight = weight;
}

//getter and setter for URL
public String getURL() {
return URL;
}
public void setURL(String uRL) {
URL = uRL;
}

//getter and setter for content
public String getContent() {
return Content;
}
public void setContent(String content) {
Content = content;
}

//getter and setter for database ID
public String getDatabaseID() {
return DatabaseID;
}
public void setDatabaseID(String databaseID) {
DatabaseID = databaseID;
}

//searchTitle is invoked when a user wants to search for a term within the articles' title
```

```
        //takes in the list of articles and the term inputed by the user
        public void searchTitle(List<Article> itemList, String inputedTerm){

        //changes the inputed term to lowercase so that the user's casing does not affect the
search
        String term = inputedTerm.toLowerCase();

        //for loop iterates through each article in the database
        for (int i = 0; i<itemList.size(); i++) {

        //searchTitle represents the title of the article currently being examined (in lowercase)
        String searchTitle = itemList.get(i).getTitle().toLowerCase();

        //assigns each "match" to an array list based on its weighting
        if (searchTitle.contains(term) && itemList.get(i).getWeight().contains("-2")) {
                farLeftList.add(itemList.get(i));
        }
        else if (searchTitle.contains(term) && itemList.get(i).getWeight().contains("-1")){
                leftList.add(itemList.get(i));
        }
        else if (searchTitle.contains(term) && itemList.get(i).getWeight().contains("0")){
                centerList.add(itemList.get(i));
        }
        else if (searchTitle.contains(term) && itemList.get(i).getWeight().contains("1")){
                rightList.add(itemList.get(i));
        }
        else if (searchTitle.contains(term) && itemList.get(i).getWeight().contains("2")){
                farRightList.add(itemList.get(i));
        }


        }
        //if all weighted arraylists are empty, alert the user the term could not be found
        if (farLeftList.size() == 0 && leftList.size() == 0 && centerList.size() == 0 &&
rightList.size() == 0 && farRightList.size() == 0) {
        System.out.print("No articles contain " + term);
        }
        //otherwise,the term could be found, calls the combineArticles method
        else {
        combineArticles(term);
        }

        }

//searchBody is invoked when a user wants to search for a term within the articles' body
//takes in the list of articles and the term inputed by the user
public void searchBody(List<Article> itemList, String inputedTerm){

        //changes the inputed term to lowercase so that the user's casing does not affect the
search
        String term = inputedTerm.toLowerCase();

        //for loop iterates through each article in the database
        for (int i = 0; i<itemList.size(); i++) {

        //searchBody represents the title of the article currently being examined (in lowercase)
        String searchBody = itemList.get(i).getContent().toLowerCase();

        //assigning each "match" to an array list based on its weighting
        if (searchBody.contains(term) && itemList.get(i).getWeight().contains("-2")) {
                farLeftList.add(itemList.get(i));
        }
        else if (searchBody.contains(term) && itemList.get(i).getWeight().contains("-1")){
                leftList.add(itemList.get(i));
        }
        else if (searchBody.contains(term) && itemList.get(i).getWeight().contains("0")){
```

```
			centerList.add(itemList.get(i));
		}
		else if (searchBody.contains(term) && itemList.get(i).getWeight().contains("1")){
			rightList.add(itemList.get(i));
		}
		else if (searchBody.contains(term) && itemList.get(i).getWeight().contains("2")){
			farRightList.add(itemList.get(i));
		}

		}
		//if all weighted arraylists are empty, alert the user the term could not be found
		if (farLeftList.size() == 0 && leftList.size() == 0 && centerList.size() == 0 &&
	rightList.size() == 0 && farRightList.size() == 0) {
		System.out.print("No articles contain " + term);
		}

		//otherwise,the term could be found, calls the combineArticles method
		else {
		combineArticles(term);
		}

		}

		//combineArticles creates an arrayList of articles across weights
		//takes in the search term
		public void combineArticles(String term) {

		int size =0;
		Article farLeftArticle = null ;
		Article leftArticle = null;
		Article centerArticle = null;
		Article rightArticle = null;
		Article farRightArticle = null;
```

```
}

//if the farLeftList has at least one article in it
if (farLeftList.size() > 0) {
//generate a random article from the list
int farLeftNum = ThreadLocalRandom.current().nextInt(0, farLeftList.size());
//set the randomly generated article to a variable
farLeftArticle = farLeftList.get(farLeftNum);
}

//if the leftList has at least one article in it
if (leftList.size()>0) {
//generate a random article from the list
int leftNum = ThreadLocalRandom.current().nextInt(0, leftList.size());
//set the randomly generated article to a variable
leftArticle = leftList.get(leftNum);
}

//if the centerList has at least one article in it
if (centerList.size()>0) {
//generate a random article from the list
int centerNum = ThreadLocalRandom.current().nextInt(0, centerList.size());
//set the randomly generated article to a variable
centerArticle = centerList.get(centerNum);
}


//if the rightList has at least one article in it
if (rightList.size() > 0) {
//generate a random article from the list
int rightNum= ThreadLocalRandom.current().nextInt(0, rightList.size());
//set the randomly generated article to a variable
rightArticle = rightList.get(rightNum);
}
```

```
        //if the rightList has at least one article in it
        if (farRightList.size() > 0) {
        //generate a random article from the list
        int farRightNum = ThreadLocalRandom.current().nextInt(0, farRightList.size());
        //set the randomly generated article to a variable
        farRightArticle = farRightList.get(farRightNum);
        }


        //if there are "matches" in each weight
        if (farLeftArticle != null && leftArticle != null && centerArticle != null && rightArticle !=
null && farRightArticle != null) {

        //add each randomly generated article to the combinedList
        combinedList.add(farLeftArticle);
        combinedList.add(leftArticle);
        combinedList.add(centerArticle);
        combinedList.add(rightArticle);
        combinedList.add(farRightArticle);

        //set size equal to the size of the combinedList
        size = combinedList.size();

        //send the combinedList and size to the outputArticles method
        outputArticles(combinedList, size);

        }

        //if there is a "match" on the left, center, or right, but no match in the far Left or farRigh
        if (leftArticle != null && centerArticle != null && rightArticle != null && (farLeftArticle ==
null || farRightArticle == null)) {
        //add the randomly generated articles from the left, center, and right
        combinedList.add(leftArticle);
        combinedList.add(centerArticle);
        combinedList.add(rightArticle);

        //size is equal to the size of the combinedList
        size = combinedList.size();

        //send the combinedList and size to the outputArticles method
        outputArticles(combinedList, size);
        }

        //if there is a "match" on the farLeft, center, and farRight, but no match in the left or right
        if (farLeftArticle != null && centerArticle != null && farRightArticle != null && (leftArticle
== null || rightArticle == null)) {

        //add the randomly generated articles from the left, center amd right
        combinedList.add(farLeftArticle);
        combinedList.add(centerArticle);
        combinedList.add(farRightArticle);

        //size is equal to the size of the combinedList
        size = combinedList.size();

        //send the combinedList and size to the outputArticles method
        outputArticles(combinedList, size);
        }

        //if there is a match in the center, but no match in the farLeft, left, right, farRight
        if (centerArticle != null && (farLeftArticle == null || farRightArticle == null) &&(rightArticle
!= null || leftArticle == null)) {

        //add just the randomly generated center article to the combined list
```

```
	combinedList.add(centerArticle);

	//size is equal to the size of the combinedList
	size = combinedList.size();

	//send the combinedList and size to the outputArticles method
	outputArticles(combinedList, size);
	}

	}

	//output articles method displays the Articles to the user
	//takes in the combined list and the int size
	public void outputArticles(List<Article> combinedList, int size) {

	//randomize the combinedList
	Collections.shuffle(combinedList);

	//Alert the user the articles are "loading"
	System.out.println ("Loading " + combinedList.size() + " aricle(s) for you to read. \nThis
is a non-biased selection.");

	try
	{
	//set the thread to sleep to slow execution for better user experience (so content does
generate so fast)
	Thread.sleep(3000);
	}
	catch(InterruptedException ex)
	{
	Thread.currentThread().interrupt();
	}
```

```
        //for loop traverses each article in the combined list
        for (Article art: combinedList) {

        //for each article that is not the last in the list
        if (art != combinedList.get(combinedList.size()-1)) {

                //Wrap the string so the article body is not printed on one line
                @SuppressWarnings("deprecation")
                String wrappedContent = ~~WordUtils~~.~~wrap~~(art.getContent(),150);

                //print the article body
                System.out.println(wrappedContent);

                //ask the user if they would like to view the source or move to the next article
                System.out.println("\nLooks like you finished reading. If you haven't, scroll back
up for the article. Press 1 to reveal the source & the next article. Press 2 for just the next article.
When the next article loads, your screen is positioned at the bottom of the article. Scroll up to
start from the begining and to see the source (if requested).");
                int nextStep = scanner.nextInt();

                //if the user chooses to view the source
                if (nextStep == 1) {
                        //print the author, title, source, and its URL
                        System.out.println("Author: " + art.getAuthor() + " Headline: " +
art.getTitle() + " Source: "+ art.getSource() + " URL: "+ art.getURL());
                        try
                        {
                        //set the thread to sleep to slow execution to give the user time to read the
source

                        Thread.sleep(4000);
                        }
                        catch(InterruptedException ex)
                        {
                        Thread.currentThread().interrupt();
                        }

                        //prepare the user the next article is coming
                        System.out.print("Your next article is loading\n\n");

                        try
                        {
                        //set the thread to sleep to slow execution for better user experience (so
content does generate so fast)
                        Thread.sleep(3000);
                        }
                        catch(InterruptedException ex)
                        {
                        Thread.currentThread().interrupt();
                        }
                        //Tell the user the next article is below
                        System.out.println("Here's the next article\n\n");
                }
        }
        //if the article is last in the array List
        else {

                //wrap the content so it doesn't print on one long string
                @SuppressWarnings("deprecation")
                String wrappedContent = ~~WordUtils~~.~~wrap~~(art.getContent(),150);

                //print the article content
                System.out.println(wrappedContent);

                //ask the user if they would like to view the article's source
                System.out.println("\nLooks like you finished reading. Press 1 to reveal the
source.");
```

```
                    int nextStep = scanner.nextInt();

                    //if they choose to view the source, print the author, source, and URL
                    if (nextStep == 1) {
                    System.out.println("Author: " + art.getAuthor() + " Headline: " + art.getTitle()+ "
Source: "+ art.getSource() + " URL: "+ art.getURL());
                    }
          }

          }
          }


}
```

<u>Database</u>

A copy of the news articles database can be found at the following link:

https://docs.google.com/spreadsheets/d/1A9zfZv0pWGebxm1FU8fopNVODsqtSlWupxqjzDHvonE/edit?usp=sharing