Web-scraping Popular News Websites for Authors' Twitter Handles

Amy Pandit
Atul Prakash, Roya Ensafi

2 May 2019

Abstract

News bias in the current sociopolitical climate has created a multitude of developments in the technological space: With advancements in machine learning algorithms and the increase in use of social media, there is a dearth of "neutral" and unbiased information for consumption by the general public. Our team aimed to look at the interaction between news articles' authors and their audiences on Twitter. My role was to web-scrape news articles and get information on the authors, including their Twitter handles, resulting in a dataset with over 600 instances. In this report, I also elaborate on the intricacies of the web-scraping process, and potential future work with news websites.

1. Introduction

With news bias comes a network of social "influencers" and opinion-shaping that occurs almost entirely untracked. This ends up creating bipartisan extremities between different news sources and groups of people, as well as selective information based on these audiences. For example, does one politician with many followers spread their information into one specific cluster (i.e. conservative) versus a talk show host that spreads their information to another cluster (i.e. liberal)?

Past research (See Section 5: Related Work) has looked into grouping different news websites based on differing ideologies on Twitter[1], as well as finding ways to illustrate Twitter topic-networks and existing clusters among users[2,3]. However, there has yet to be research about web-scraping news websites and using that data alongside Twitter research. There has also been little to no specific research on the authors who write news articles, and whether their handles could be of use in analyzing bias.

Our proposed solution to determining media bias involves using information retrieval techniques for both Twitter data and news websites. We can begin to see how information appearing in different sources is circulated in social networks. These steps are done by extracting Twitter data from selectively chosen influential users, and extracting their tweets and the news links they reference, seeing who follows them and interacts with their tweets, and building a network. There also needs to exist a system of storing websites and their content, and also builds a network based on the content that website references or redirects to. Moreover, the networks we build based on these two extraction points will better inform our future analysis of media bias.

My role in this project was focusing on web-scraping news websites' home pages and getting information about both the article and author -- When was the article written, by whom, and do they have a Twitter handle? This last piece of the puzzle was by far the most important, as ultimately it will be used in the Twitter network analysis bulk of the project. Keeping track of the article information was also an important addition, as we could find a way to figure out the content and subject of an article to use in analysis of Twitter responses or retweets for that subject. A lot of my work was exploratory in nature, and I'll be going through my process of web-scraping, in addition to my code infrastructure, to better inform future researchers and students working on this project about the use of news websites for Twitter network analysis.

The results of this web-scraping was 600+ instances of web-scraped article data with author Twitter handles, along with insight on how news webpages are structured. Thinking ahead about storage plans after scaling this data was also an important step, and I looked into the Google Cloud Platform as a potential solution. Finally, I conclude that perhaps research on this web-scraping would be better spent looking at whether authors publicize their Twitter account, and if there are any existing patterns or relationships based on someone's Twitter not being linked to the article they write for a particular news site.

2. Methods

An overview of my code's functionality is as follows:

1. Visit home page of the news website
    a. Take a screenshot
    b. Strip all article links on that home page only
2. Save all current articles to a directory with their raw HTML before any cleaning is done
3. Visit each article
    a. Find where the Twitter handle is located as well as other information like date of publication, author name, article title, site name, category, and text preview (which is just a plain text strip of the entire article)
    b. If Twitter handle can't be found, look if they have a profile link, and if so, visit it and find the Twitter handle there
4. Only add to the database where a Twitter handle is found with its corresponding author name (this restriction can be changed later)
5. Save the table (article_data) as a .csv file to then upload to Google Cloud

Below, I get into more specifically which technology was used to accomplish these tasks.

2.1. Technologies Used

I started by using a virtual Python environment, using the default Python 2.7.4 on my Mac OSX. Running this code on Python 3 was a major troubleshooting issue for me, as my local system had some dependencies and/or version issues; however, most of this code can be migrated to Python 3 fairly easily, with the exception of a couple of packages that are made available for Python 3 from third-party sources (i.e. Html2Text, Selenium). The following notable packages are what I used throughout this project:

1. Selenium
    - Webdriver was used to take a screenshot of the front page of each news website and store it in its appropriate folder locally. This code could be useful in the future for storing multiple versions of this screenshot; for my project, I just kept updating the same file.
2. BeautifulSoup4
    - Used for HTML extraction of entire webpages. This package is also able to extract specific tags based on attributes or class names.
3. Re
    - Using regular expressions in Python was key for web-scraping; for future reference, use them when you cannot use BeautifulSoup. Regular expressions became helpful when BeautifulSoup proved to have limitations about what it could search for. I tended to use regular expressions because of the messiness of HTML; however, it does take a bit longer to run, which is a scalability issue, discussed under "Scalability Discoveries".
4. Html2Text
    - Used to take HTML and clean it up so that the content in the database stored for each article was clear of most formatting. This will make it easier to parse and for a possible sentiment analysis extension.
5. Sqlite3
    - SQLite was a way to test code locally and save it to a database on my personal system. This database contained a table, article_data, that contained the web-scraped results.
6. Google.cloud
    - Google Cloud Platform makes it possible to run Python code through its online application, but Python also has a package that makes it possible to connect to Google Cloud locally and upload "objects", for example, a CSV file with the data dump from the web-scraping.
7. articleDateExtractor
    - Maintained by Webhose.io, this module has a fairly high accuracy of extracting the date if it's made publicly available on an article. (Over a 90% success rate)[4]

2.2. Process

I began with implementation with CNN alone, and then added Fox News. Because of this, the initial blocks of code in most parts of the extraction.py (which is the main file with all of the extraction and insertion into database) were specific to CNN. I structured the article_data SQLite table with the following schema:

```
CREATE TABLE article_data (
article_link VARCHAR NOT NULL,        # full link to the article
article_date VARCHAR,                 # extracted date to the article
author_name VARCHAR NOT NULL,         # full name of author (if multiple authors, will
                                        appear in separate entries in table)
author_handle VARCHAR NOT NULL,       # author's Twitter handle
site VARCHAR NOT NULL,                # site key (ex: cnn, foxnews, thehill)
article_title VARCHAR,                       # title of the article
category VARCHAR,                                    # category of the article
text_preview VARCHAR,                        # a cleaned version of stripped text from article
PRIMARY KEY (article_link, author_handle)            # this asserts that the article link and author
);                                                     handle pair will be unique
```

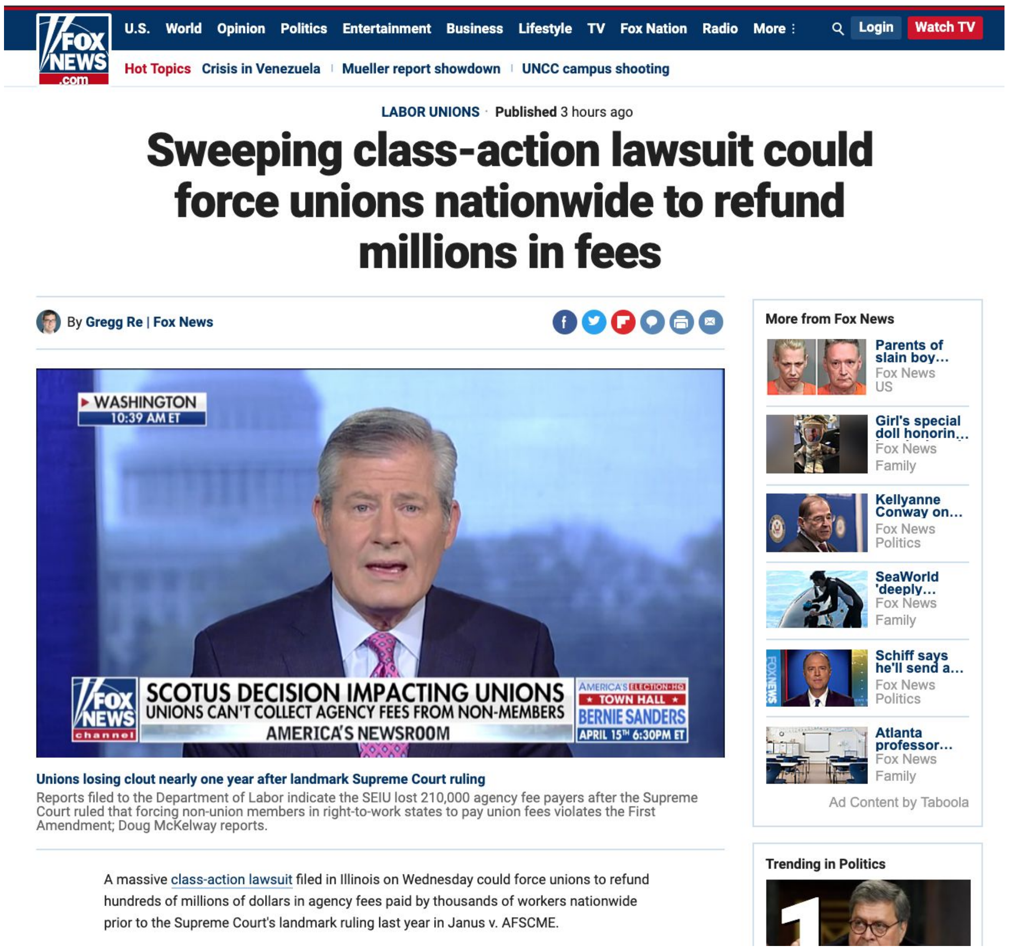
After adding the Fox News implementation to compare with CNN, I began to take a more holistic approach and think about how the code would scale to multiple news websites. This then created implementation with the idea of “footprints”, where specific patterns of extracting the Twitter handle and other information about the article manifested in various types of if-statements. This made the code more versatile rather than focusing on one new website at a time. I did research on ~20 of the top news websites, including but not limited to: Yahoo! News, The Hill, Time, CNBC, The New York Times, and ABC News. After observing the raw HTML, I was able to come up with a handful of these “footprints” to try to “automate” extraction, without having to cater to each specific website (For code infrastructure, see Appendix A1).

3. Results

An exploratory project by nature, many of the results of my work ended up being related to web-scraping practices and figuring out how feasible it was to create a more automated process of extraction across multiple different sites at once. There were also some observations made about the qualitative data and perceived bias. The following results show that it is possible to create a system of automatically scraping news websites (though it is more complex and requires larger systems) and that there are results that are promising for future work in media bias.

3.1. The Data

The following is an example of an article found on the home page of Fox News.


U.S. World Opinion Politics Entertainment Business Lifestyle TV Fox Nation Radio More
Login
Watch TV
Hot Topics Crisis in Venezuela | Mueller report showdown | UNCC campus shooting
More from Fox News
Parents of slain boy... Fox News US
Girl's special doll honorin... Fox News Family
Kellyanne Conway on... Fox News Politics
SeaWorld 'deeply... Fox News Family
Schiff says he'll send a... Fox News Politics
Atlanta professor... Fox News Family
Ad Content by Taboola
Trending in Politics
WASHINGTON 10:39 AM ET
SCOTUS DECISION IMPACTING UNIONS
UNIONS CAN'T COLLECT AGENCY FEES FROM NON-MEMBERS
AMERICA'S NEWSROOM
AMERICA'S ELECTION HQ ★ TOWN HALL ★ BERNIE SANDERS APRIL 15TH 6:30PM ET

LABOR UNIONS · **Published** 3 hours ago

Sweeping class-action lawsuit could force unions nationwide to refund millions in fees

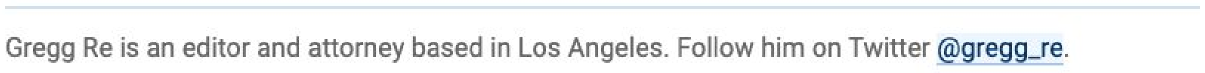
By **Gregg Re | Fox News**

Unions losing clout nearly one year after landmark Supreme Court ruling

Reports filed to the Department of Labor indicate the SEIU lost 210,000 agency fee payers after the Supreme Court ruled that forcing non-union members in right-to-work states to pay union fees violates the First Amendment; Doug McKelway reports.

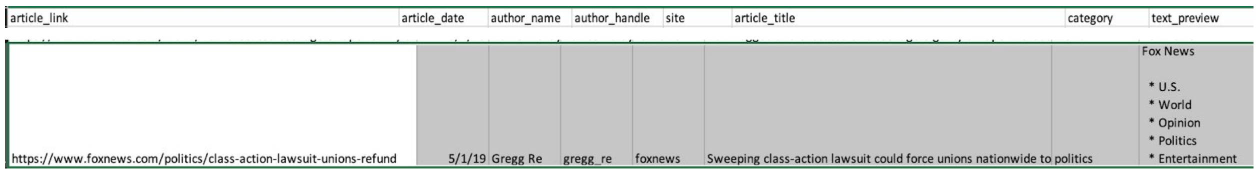
A massive class-action lawsuit filed in Illinois on Wednesday could force unions to refund hundreds of millions of dollars in agency fees paid by thousands of workers nationwide prior to the Supreme Court's landmark ruling last year in Janus v. AFSCME.

At the bottom of the page, the Twitter handle happens to be stated explicitly, without having to visit an additional author profile page (though this may be required on certain articles and especially on other news sites aside from Fox News).



Gregg Re is an editor and attorney based in Los Angeles. Follow him on Twitter @gregg_re.

Using regular expressions directly on the HTML page itself, these Twitter handles for Fox News can be easily extracted and cleaned, if necessary. An example of this article in the database itself is as follows:



article_link	article_date	author_name	author_handle	site	article_title	category	text_preview
https://www.foxnews.com/politics/class-action-lawsuit-unions-refund	5/1/19	Gregg Re	gregg_re	foxnews	Sweeping class-action lawsuit could force unions nationwide to	politics	Fox News * U.S. * World * Opinion * Politics * Entertainment

Note that the text_preview column includes all text, including the website headers that appear at the top of every page.

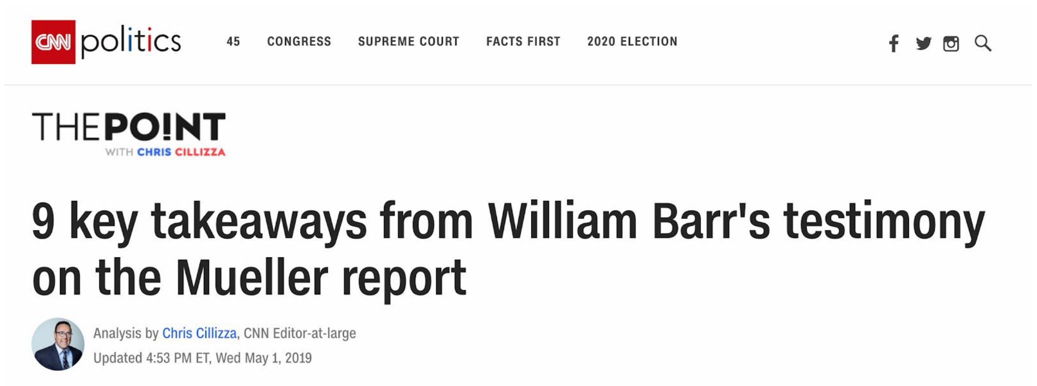
In this project I've collected 631 total articles and their corresponding author Twitter handles. I've attached the data in article_dump.csv along with this report.

3.2. Differences in Web-page Setups

By far the greatest result from this project was seeing how different news websites were formatted once the HTML was stripped. Though the idea of automation was appealing at first and seemed straight-forward, seeing how different news sources formatted their HTML made the task a bit more complex than initially anticipated.

For example, with the first task of finding all the articles on the home page, sometimes there was a specific HTML tag that was easy to grab, and other times, *all* links on the page had to be taken (Refer to the example of the difference in this code in Appendix A2). In the case that all links were taken, by checking for Twitter handles of authors, it was possible to prevent any non-articles from being added to the database.

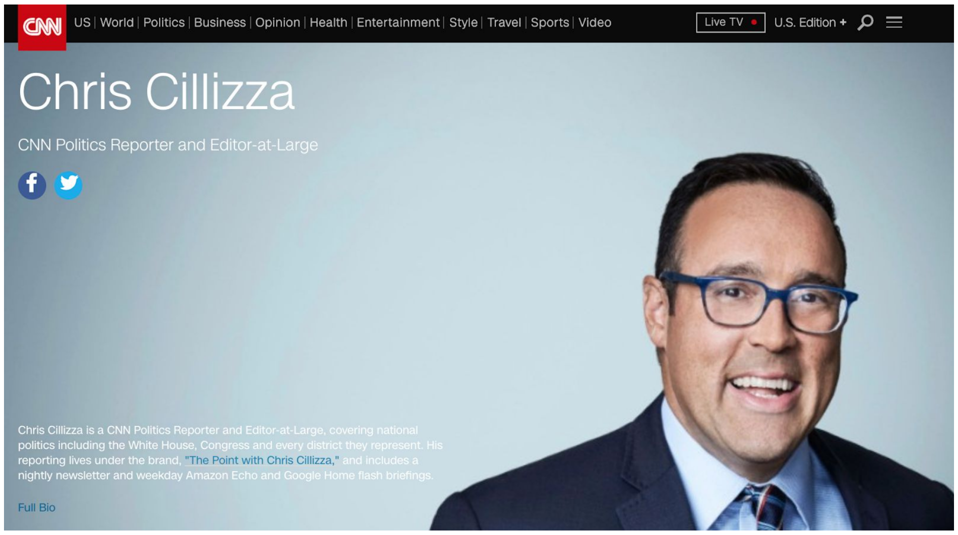
Secondly, finding the Twitter handle was another complicated task. As shown in the example in 3.1, Fox News was one website that explicitly stated "Follow [author] on Twitter @[handle]". This format meant code could be written to specifically find this terminology to extract the handle. Another common setup shared by some news websites was the following:



9 key takeaways from William Barr's testimony on the Mueller report

Analysis by Chris Cillizza, CNN Editor-at-large
Updated 4:53 PM ET, Wed May 1, 2019

Here, there is no pattern as seen above as the handle explicitly being listed on the article. However, once you click on the author's name (in this case, Chris Cillizza), you are redirected to their profile page:


CNN
US | World | Politics | Business | Opinion | Health | Entertainment | Style | Travel | Sports | Video
Live TV
U.S. Edition +

Chris Cillizza

CNN Politics Reporter and Editor-at-Large

Chris Cillizza is a CNN Politics Reporter and Editor-at-Large, covering national politics including the White House, Congress and every district they represent. His reporting lives under the brand, "The Point with Chris Cillizza," and includes a nightly newsletter and weekday Amazon Echo and Google Home flash briefings.

Full Bio

The Twitter icon links to their Twitter account, and can be extracted this way. This adds another page request, so this method should only be used if the handle is not found anywhere else. For CNN, this was the only way to get the handle.

In summary, the results from this web-scraping process were as follows:

1. Take time to look at website infrastructure *before* beginning implementation of the web-scraper. I started with CNN and made it extremely specific to that site, and then looked into other websites to compare. If I had done this research initially, there would not be an "if site_key == 'cnn'" portion of my code.
2. If the Twitter handle is on the article page, try to find that *before* opting to find an author profile page. Preventing requests to websites will speed up the process.
3. In all cases, start specific, and then move to general. It's not efficient to find all the <a href> links on a page first, before trying other footprints, since there are many links on a homepage that are not articles.
4. Not every coding problem can be perfectly scalable. After seeing all the different HTML formats for different websites, it was clear that front-end development varied greatly, to the point where automation almost seemed infeasible. I do, however, believe it's possible for this particular problem to scale (See Section 3.4).

3.3. Fox News Twitter Handles' Success Rate

As I added Fox News to my implementation following CNN, I did notice that there were significantly fewer results of handle extraction. This was not because the code was failing; by manually checking, it showed that the authors' Twitter handles were in fact not appearing on the articles they wrote. Because it seemed interesting that Fox News appeared to have far fewer instances of Twitter handles than CNN did, I went through manually to try to see if these authors lacking handles actually did have Twitter accounts that just weren't publicized.

The results were interesting - many of the authors extracted who did not have handles listed on Fox News actually did have very active Twitter accounts. Some were private, but most were very clearly politically-driven accounts. Looking at the first 20 results of the top news articles on the front page of Fox News, where the author name was found but had no handle, I found that 11 of the top authors extracted actually *did* have Twitter handles. These handles, however, were not publicly listed on any profile page or article on Fox News. This is interesting, because even after manually examining the Twitter accounts, very few were listed as private (in fact, for this small sample, only 1 account was private). The rest were public, with varying degrees of political activity. Some of the authors did not even acknowledge that they wrote for Fox News.

This brings up a qualitative point in our analysis of finding Twitter handles -- perhaps the authors of more conservative news are less likely to publicly advertise their conservatism, compared to their liberal counterparts. For CNN, manually checking reveals that almost every single political author with a Twitter handle has it listed on their profile. That being said, there is the consideration that a lot of the authors on Fox News were not necessarily exclusively writing for Fox News; for example, some actually wrote for the NY Post and advertised that on their Twitter accounts instead of Fox News. These differences can all contribute to the likelihood that a website will list an author's Twitter handle on an article they write.

3.4. Scalability Discoveries

This project was important for realizing how scalable a project like this could be. Given that I was running this code locally with my database in SQLite, I noticed that it didn't seem feasible to try to scale this to more than about 6 websites (running this code with 8GB RAM on my Mac would take about 15 minutes). With anywhere from 40 to almost 100 links per home page, trying to scrape through the article or even the author's profile page to find the handle was too intense of a process for a local machine. Though there definitely exist ways to make my current code more efficient, this task is still too heavy, and because of the variability across news websites, the code could get long and ugly.

This was unsurprising, of course, and I looked into the Google Cloud Platform as a potential application builder, which could run the code on its platform directly. This, along with other applications of Google Cloud, are discussed in the next section.

Ultimately, I believe this project is valuable and would require more memory and more efficient storage techniques. In addition to using Google Cloud Platform for running the scripts and storing the data as described in Section 3.1, my code currently also saves the raw HTML for each of the articles it scrapes. However, this directory of files deletes itself before each run, to prevent overload of my personal laptop. If this code could be developed and run on Google Cloud, saving *all* the articles for future analysis could be helpful, since the plain text in the text_preview attribute may not tell all.

3.5. Google Cloud Debugging
Throughout this project, there was a bit of debugging that I tried to do to work with Google Cloud Computing Services. Ultimately, the route I ended up taking was --

1. Run code locally to populate SQLite database
2. Run code locally to export SQLite table to a csv file, called article_dump.csv
3. Manually upload this CSV file to the mediabias bucket on Google Cloud

This is clearly not automated, however. Though I did produce code that was supposed to be used for automating this process, and also separately looked at App Engine within Google Cloud (which likely should be the next action step for this project, see Section 6.2), there were some local bugs related to authentication that were difficult to spot-fix. Because of this, I decided to move on from this debugging process in favor of continuing to work on the actual web-scraping code instead; however, in the future, code should be run and scheduled within App Engine to automatically scrape news websites on a daily basis and update the database, as well as saving any additional files or information that could be used in the future (i.e. all scraped HTML from the website, screenshot of the home page, etc.)

4. Discussion
What is the significance of these results? Given that this is an exploratory project, the primary question is: Is this part of the project worth pursuing? How does this fit in with the rest of the media bias project? Are there any extensions that we could look into in the future?

Web-scraping requires thorough observation of the types of webpages prior to writing code, *especially* if the goal is ultimately automation. Optimizing code for web-scraping is important, but the issue of scalability always exists given the sheer volume of types of websites and the level of variability that each brings. I believe that looking at these news websites and looking at scraping their information is useful, but I believe the interesting point is not necessarily related to the Twitter networks as we originally saw the problem.

Originally, we thought about just using the web-scraping to extract Twitter handles to add to our networks, with potentially using the content of the article for sentiment analysis, or checking who shared that specific link, and what their comments on it was. This is definitely a possible route for the media bias project, but I think the complexity of web-scraping isn’t worth just looking for these components.

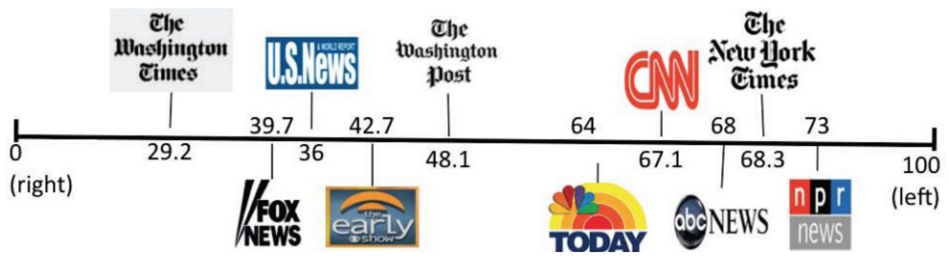
Resources would likely better be spent if we used this web-scraper to look at the correlation between how likely authors are to publicize their Twitter handle, what content they’re writing articles about, whether they’re liberal or conservative -- and then comparing this with their public persona on Twitter. I came across this potential route a little later in the process, so I could not look into it very thoroughly, but even with the results from Section 3.3, I think it’s possible that authors who do not publicize their Twitter handles could be linked to their political beliefs. Furthermore, there could be a difference between the audience of their article and the audience of their tweets, for example. This is a proposed hypothesis for future work.

Thus, I believe that research for the news website portion, if it continues, should focus more on finding authors who do *not* reveal their Twitter handle, but still have one that exists. This is a little trickier of a problem, since there needs to be a way to validate an author's name and their Twitter handle from outside sources or a Google search, which could be infeasible if someone's name is very common. Regardless, I do think this is an interesting problem that pertains to media bias, because it shows the differences between someone doing their job (i.e. writing an article for a website who pays them) and their Internet persona (i.e. their personal opinions and tweets to their followers). If this research is done, it is important to realize that these authors may not be *intentionally* hiding their handle from their article; for example, perhaps Fox News just does not update as much as CNN does. That being said, it seems that people treat articles and tweets very differently in the political sphere, and it would be interesting to see more research about that relationship.

5. Related Work

Himelboim, et al.[2] worked on a study that classified Twitter topic-networks using social network analysis, which is pretty similar to what we would be implementing, except focusing on political news. While they observed 6 different "hashtag" topics, our work would be a little different, instead evaluating how a specific piece of news can spread across the social network. This study is important in understanding divided, polarized (high graph density) and fragmented, unpolarized (low graph density) clusters, and identifying other patterns of information flow on Twitter. Related to this, Wong, et al.[3] brought in the political aspect of the analysis, observing sentiment of tweets and assigning a "political leaning score" based on predictors. They also looked at the differences in the tweets of political candidates, media sources, and parody accounts. This adds another dimension to media bias research that's valuable to take into account when conducting our own studies; there are many score-calculating models out there, so it doesn't seem like that is something that needs to be reproduced.

An, et al.[1] worked on visualizing media bias through Twitter, and displayed the graphic below as part of their results:



The numbers are coordinates, and the closeness of the news sources with one another are based on similarities; i.e. ABC news is closest to The New York Times, and typically the left-leaning news sources are clustered together (in this image, on the right side), and the right-leaning are clustered on the left side. Again, these were based on tweets, and not the news websites themselves; however, this shows that other people have begun to look at the bias among different websites. Our project can take this a step further and try to bridge the gap between what the news publishes, and how that connects to information networks on Twitter.

6. Conclusion

6.1. Summary

Retrospectively, I found that web-scraping required a deeper understanding and more thorough observation of the source code for different news websites. The system for developing this web-scraper should also build footprints starting from more specific to more general, in order to be as accurate as possible while making the fewest number of requests. This web-scraping problem is also scalable, but would need to be placed on a cloud platform, likely Google Cloud. There were 631 total articles with corresponding Twitter handles that were extracted, and these can be used for future analysis if needed; however, I do have recommendations for future work in the following section.

6.2. Future Work

[which functions you want to add in the future, and why, what you'd want to change about how your code works, what you want to keep, HOW TO SCALE, combining with bias analysis of tweets in the future, getting rid of CNN-specific code and transitioning to general, etc.]

For my web-scraping code in extraction.py, I would like to get rid of the CNN- and Fox News-specific code, and just integrate those into the footprints for other websites. I'd also like to modify some of the functions to make them more independent of each other. Right now, find_articles() is calling write_articles(), which then calls the primary web-scraping function, insert_authors_articles(). This is an unnecessary chain that could be split up potentially to make future testing easier. For example, saving a variable after find_articles(), then passing that into write_articles() in a separate function call in main(), and so on. Also, there were a few lines of repeated code for cleaning a handle or a list of articles, and these could be placed in their own special functions.

Moreover, for using Google Cloud, checking the Google documentation[5] for AppEngine would be helpful for scaling in the future. In addition, crontab is a potential Python package that I was initially looking into to automate scripts, so I could run my web-scraper once a day, for example. If automation is important in future iterations of this web-scraping problem, then I would recommend seeing if this package is usable. I would also switch over completely to Python 3.7, since though it was my intention to switch over at the beginning of the project, the issues I was running into while debugging seemed to be a deeper system issue with SSL certificates. If any of this code is to be run in the future on version other than Python, it is possible that some packages may need to be reinstalled from another source, or substituted. I would suggest looking into Newspaper3k as well, since it seemed to have a lot of useful functionality for extracting information from news articles (ones that I did not necessarily need for the scope of my project, but could be helpful in future analysis work).

Additionally, based on my proposed future hypothesis in (See Section 4: Discussion), the database should be modified to not require the Twitter handle in the article_data table. This would simply involve the handle not being part of a primary key, since the scope of the project would be changing a bit, meaning that articles with no handles listed would be important as well. This change in direction is merely a suggestion, but would hopefully give the web-scraping problem more depth and importance than only taking handles from news articles.

References

1. An, J., Cha, M., Gummadi, K.P., Crowcroft, J., & Quercia, D. Visualizing Media Bias through Twitter. 2012. *AAAI Technical Report WS-12-01, The Potential of Social Media Tools and Data for Journalists in the News Media Industry.* pp. 2-5.
2. Himelboim, I., Smith, M. A., Rainie, L., Shneiderman, B., & Espina, C. Classifying Twitter Topic-Networks Using Social Network Analysis. January-March, 2017. *Social Media + Society*. pp. 1-13.
3. Wong, F. M. F., Tan, C. W., Sen, S., & Chiang, M. Quantifying Political Leaning from Tweets and Retweets. 2013. *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media.* pp. 640-649.
4. https://github.com/Webhose/article-date-extractor
5. https://cloud.google.com/appengine/docs/standard/python/quickstart

Appendix

A1. Code infrastructure

extraction.py
Summary: This contains all the code for extraction from news websites and saving to the database. This could potentially be split up into multiple files in the future for easier testing, scaling, and readability.

Important functions:

- **`display_article_data(conn)`**
  - `# Display article_data table from the database`
- **`is_valid_url(url)`**
  - `# Checks that a url is valid and does not return an error`
- **`is_valid_handle(handle)`**
  - `# Checks that a Twitter account for the given handle exists`
- **`save_screenshot(url, direct_site, site_key)`**
  - `# Takes a screenshot of home page, saves to a file location`
- **`get_preview(soup_article)`**
  - `# get plain text of an article`
- **`save_homepage(url, site_key)`**
  - `# Create local directories for a given site`
- **`write_articles(url, articleLinks, site_key, folder, cur, conn)`**
  - `# Takes a list of article links, loops through and saves HTML in a text file for each`
- **`find_articles(url, site_key, soup, cur, conn)`**
  - `# Scrapes the homepage and finds all (potential) article links`
- **`insert_authors_articles(url, site_key, articleLinks, cur, conn)`**
  - `# Scrapes all the information from each of the articles and adds to database`

test_footprints.py
Summary: This was a test file that was used for testing footprints; it's not important to the final result, but I included it to give an example of how testing occurred.

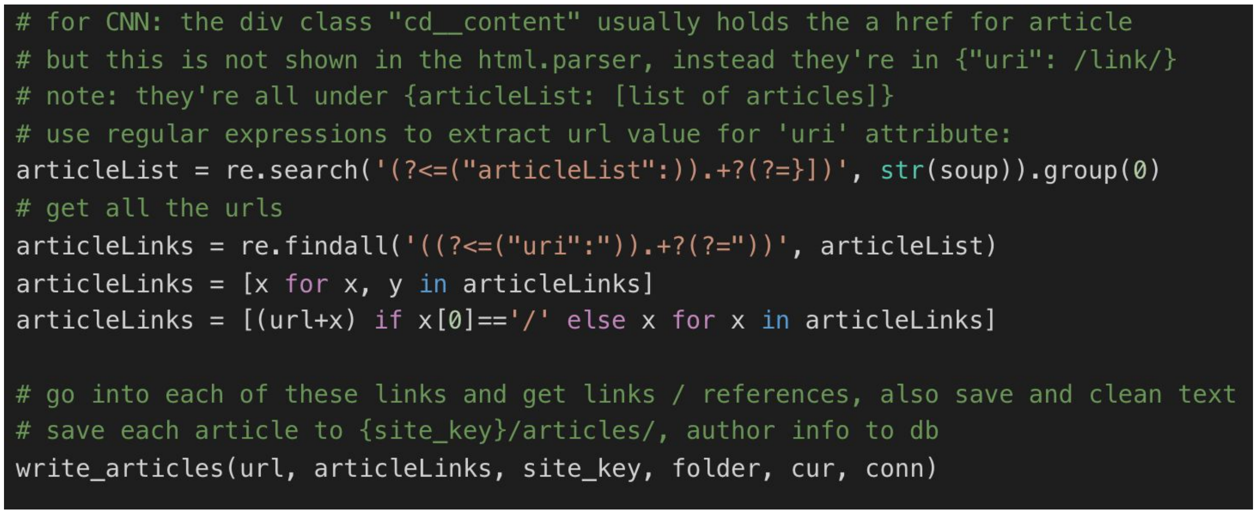
dumpdata.py
Summary: This contains important code that is commented out -- I used it for opening sqlite and saving the extracted data to article_dump.csv. This could've also been put in a bash script. There is also a Google Cloud function declared with some usage also written in a main(), but it wasn't

used because of the issues with debugging Google Cloud. They could be tested in the future and used for automation, in addition to the sqlite code.

- **`upload_blob(bucket_name, source_file_name, destination_blob_name)`**
  - **`Source:`** [`https://cloud.google.com/storage/docs/uploading-objects#storage-upload-object-python`](https://cloud.google.com/storage/docs/uploading-objects#storage-upload-object-python)

A2. Differences in finding all article links on the home page (extraction.py)

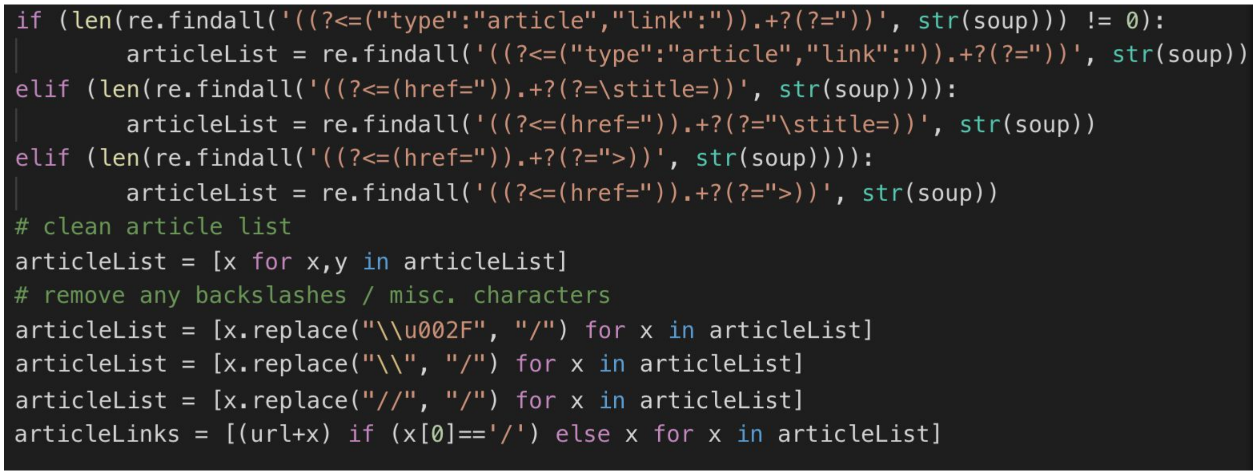
CNN: all the article links were already in a list embedded in the HTML. This made it easy to extract, knowing that all of these links were articles.



```python
# for CNN: the div class "cd__content" usually holds the a href for article
# but this is not shown in the html.parser, instead they're in {"uri": /link/}
# note: they're all under {articleList: [list of articles]}
# use regular expressions to extract url value for 'uri' attribute:
articleList = re.search('(?<=("articleList":)).+?(?=}])', str(soup)).group(0)
# get all the urls
articleLinks = re.findall('((?<=("uri":")).+?(?="))', articleList)
articleLinks = [x for x, y in articleLinks]
articleLinks = [(url+x) if x[0]=='/' else x for x in articleLinks]

# go into each of these links and get links / references, also save and clean text
# save each article to {site_key}/articles/, author info to db
write_articles(url, articleLinks, site_key, folder, cur, conn)
```

Other news sites: I provided a few possible ways (from specific to general) to see if I could catch the article links in different websites. This also involved code that would clean the links if needed.



```python
if (len(re.findall('((?<=("type":"article","link":")).+?(?="))', str(soup))) != 0):
        articleList = re.findall('((?<=("type":"article","link":")).+?(?="))', str(soup))
elif (len(re.findall('((?<=(href=")).+?(?=\stitle=))', str(soup)))):
        articleList = re.findall('((?<=(href=")).+?(?="\stitle=))', str(soup))
elif (len(re.findall('((?<=(href=")).+?(?=">))', str(soup)))):
        articleList = re.findall('((?<=(href=")).+?(?=">))', str(soup))
# clean article list
articleList = [x for x,y in articleList]
# remove any backslashes / misc. characters
articleList = [x.replace("\\u002F", "/") for x in articleList]
articleList = [x.replace("\\", "/") for x in articleList]
articleList = [x.replace("//", "/") for x in articleList]
articleLinks = [(url+x) if (x[0]=='/') else x for x in articleList]
```

A3. Main() function (extraction.py)

```
def main():
        # use this connection for SQL queries
        conn = sqlite3.connect('extraction_db.sqlite')
        conn.text_factory = str
        cur = conn.cursor()

        # define sites and their site "keys": these keys are important for creating
directories
        # I also used these site keys as identification in the database
        sites = ['https://cnn.com', 'https://foxnews.com','https://news.yahoo.com',
'https://cnbc.com/politics', 'https://time.com', 'https://thehill.com' ]
        site_keys = ['cnn', 'foxnews', 'yahoonews', 'cnbc', 'time', 'thehill'] # site
keys are the unique ID
        # below code, commented out, is for individual testing of sites
        # site = sites[5]
        # site_key = site_keys[5]

        # loop through sites and their identifying site keys
        for idx in range(len(sites)):
                site = sites[idx]
                site_key = site_keys[idx]
                if not is_valid_url(site):
                        print(site + " cannot be accessed.")
                        continue
                # save homepage as soup variable, also save screenshot
                soup = save_homepage(site, site_key)
                # use homepage to find articles and then insert them into database
                # currently, this function calls other functions within it,
                # we can change it in the future so it calls them one at a time,
                # independent of one another
                find_articles(site, site_key, soup, cur, conn)
        # print out current state of table after insertions
        display_article_data(conn)

        # MANUAL: export data to csv through sqlite, then upload to Google Cloud bucket

        # make sure to commit so the changes to the database remain
        conn.commit()
        conn.close()
```