WebSearcher: Tools for Auditing Web Search

Ronald E. Robertson
Northeastern University
rer@ccs.neu.edu

Christo Wilson
Northeastern University
cbw@ccs.neu.edu

ABSTRACT

Independent investigations of web search engines have become more prevalent in the last decade, but tools for conducting them have not. We introduce `WebSearcher`, an open source python package for archiving and parsing search engine results pages (SERPs) from a major web search engine. It features a modular parser for decomposing a SERP into a set of components while preserving details about their positioning and formatting, which exert a strong influence on what users pay attention to. Building upon prior work that identified 14 unique components [21], we identified 22 unique components. Our goal in building this package is to lower the technical barriers to entry and enable researchers from all backgrounds and disciplines to examine the outputs of web search within the context of their own unique research interests and questions. For example, one could use this package to collect and parse data on how search engines represent politicians [16, 20], minorities [19], conspiracy theories [1, 9, 10], or health-related topics [3]. In this paper we provide step-by-step instructions for using `WebSearcher` while outlining its functionalities and limitations.

ACM Reference Format:
Ronald E. Robertson and Christo Wilson. 2020. WebSearcher: Tools for Auditing Web Search. In *Proceedings of Computation + Journalism Symposium 2020 (C+J '20).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

INTRODUCTION

There is a long history of people examining the results returned by web search engines, but only recently have these studies broadly entered public discourse and legal debate. This entry is, in part, due to growing concerns that search engines can contribute to the oppression and marginalization of minority groups [19] and have the potential to influence critical social functions like voting [7, 12]. However, it is also due to the evolution of search engine studies - not just in terms of the methods used, but also in terms of who conducts them and how they broadcast their findings.

Today, these socially critical systems are being computationally investigated not just by information retrieval scientists, but by qualitative researchers, journalists, and corporate research labs among others [5, 6, 23, 24]. Despite this rising number of web search investigations, there have been few efforts to release and maintain tools for conducting them.

One reason for the lack of open source libraries for conducting web search audits, is a technical one: the outputs of search engines are a moving target. That is, their complexity evolves over time - with knowledge graph, news, and map components all emerging in recent years [13, 15, 18] - and is further compounded by A/B testing.

Another major concern is a legal one: the legality of web scraping. Automated collection of Search Engine Results Pages (SERPs), despite their public-facing nature and their potential to influence the public [7, 8, 11, 12], could be a legally risky endeavor. Until very recently, there lacked legal precedent for how the automated collection of public-facing websites would be treated in a court of law. However, on September 9, 2019 the court ruled in *hiQ Labs, Inc. v. LinkedIn Corp.*, that accessing "publicly available data will not constitute access without authorization under the CFAA [Computer Fraud and Abuse Act]" [2]. This ruling, while not the final word, opens the door to releasing online data collection projects like ours, which we hope is the first of many.

Here we introduce `WebSearcher`, an open source python package for conducting web search audits. The goal of this work is to share the means to conduct web search audits and enable researchers from diverse background and disciplines to examine these systems through their lens. It includes tools for geolocating, scraping, and saving search data, as well as a modular parser for decomposing a SERP into list of components with categorical classifications and position-based specifications. We chose this modular design to enable ongoing updates for novel components, such as those that appear during specific seasons, including election seasons [4].

C+J '20, March 20-21, 2020, Boston, Massachusetts

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 GETTING STARTED

The latest version of `WebSearcher`[1] can be downloaded from the Python Package Index:

```
pip install WebSearcher
```

2 CONDUCT A SEARCH

To conduct a search: (1) import the package, (2) initialize a connection for sending requests to the search engine, and (3) conduct a search and receive the data.

```
import WebSearcher as ws

# Initialize search engine crawler
se = ws.SearchEngine()

# Conduct search and retrieve data
se.search('immigration')
```

3 SAVE A SEARCH

We recommend immediately saving the raw HTML of incoming SERPs before proceeding. This helps to avoid unexpected errors that could crash a search session, resulting in missing data for longitudinal efforts. Our package currently provides two formats for saving the data from a search: (1) appending it to a JSON file, or (2) saving it as an HTML file. For continuous or large scale collection, we recommend the JSON approach to avoid accumulating excessive data files. For qualitative explorations of smaller query sets, the HTML approach enables you to quickly look at the results for specific searches[2] by opening the HTML files in your browser.[3]

```
# Append to a json file
se.save_serp(append_to='serps.json')

# Save as a .html file in a selected directory
se.save_serp(save_dir='./serps')
```

4 PARSE A SEARCH

As web search continues evolving beyond 10 blue links, parsers that capture the diverse ways in which information is presented and arranged for information seekers will become crucial for understanding how users interact with these systems [4, 15, 21, 22, 25]. To parse a SERP into a JSON list of categorized components[4]

```
# Convert to a parsable format
soup = ws.make_soup(se.html)

# Parse the SERP
results = ws.parse_serp(soup)

# Example output:
results[0]
{
    'type': 'ad',
    'serp_rank': 0,
    'title': 'Aspen Ideas Festival | 21st Century US ...',
    'url': 'www.aspenideas.org/',
    'text': 'US immigration processes are convoluted ...',
    ...
}
```

5 CONDUCT A LOCALIZED SEARCH

A *localized search* is a search conducted on your computer,[5] but from a location of your choice. To conduct localized searches: (1) obtain a dataset of "canonical names" for locations using a function that obtains the most recent version,

```
# Set save location
data_dir = './location_data'

# Download latest data
locs = ws.download_locations(data_dir)
```

(2) find the canonical name for your selected location

```
f  = os.listdir(data_dir)[-1]  # Last file
fp = os.path.join(data_dir, f) # File path
locs = pd.read_csv(fp)         # Read

# Find name containing city (Boston) and state (MA)
regex = r'(?=.*Boston)(?=.*Massachusetts)'
str_mask = locs['Canonical Name'].str.contains(regex)
```

and, (3) add the canonical name to your search request.

[1] For more examples, source code, and contribution instructions, please see our Github page at: https://github.com/gitronald/WebSearcher.
[2] HTML files are named by a user-provided or randomly generated ID.
[3] The HTML approach does not store meta data about the search, although timestamps could be recovered from the files.
[4] At present, we can account for 22 unique component types, including ads.
[5] Localized searches can also be conducted from a remote computer (see §6).

```
# Conduct a localized search
loc = 'Boston,Massachusetts,United States'
se.search('immigration', location=loc)
```

6 CONDUCT A REMOTE SEARCH

A *remote search* is a search conducted on a remote computer. To connect to that computer, we use secure shell (SSH)[6], a standard software package that enables secure file transfers over the internet. In this case, the files we're transferring are our requests to a search engine, and the response we get is the resulting page of search results.

```
import subprocess

# Set SSH Connection settings
ip = XX.XXX.XX.XX
port = 6000
keyfile = '/path/to/mykey.pem'
ssh = ws.webutils.SSH(port=port, ip=ip, keyfile=keyfile)

# Set permissions on keyfile and open the SSH tunnel
subprocess.call(['chmod', '600', keyfile])
ssh.open_tunnel()

# Start a requests session on the same SSH port
sesh = ws.webutils.start_sesh(proxy_port=ssh.port)

# Collect a SERP through the remote computer
se.search('immigration', sesh=sesh, ssh_tunnel=ssh)
```

7 CRAWLING SEARCH RESULT URLS

To obtain additional granularity of the windows to different corners of the web, one can also collect the HTML for the URL associated with each result on the SERP. As with saving the SERP HTML, we offer two options here: (1) append it to a JSON file, or (2) save it as an HTML file, and again recommend the JSON approach.

```
# Set save file
fp_results_html = '/path/to/results_html.json'

# Parse SERP (inplace) to extract URLs
se.parse_serp()

# Scrape HTML from results URLs and save to file
se.scrape_results_html(fp_results_html)
```

[6]https://www.ssh.com/ssh/

If the search session was initialized with an SSH tunnel, then all of these requests will be funneled through it.

DISCUSSION

In this paper we introduced `WebSearcher`, a python package for geolocating, archiving, and parsing web search data. Our goal in releasing this package at an interdisciplinary conference is to democratize the ability to conduct web search audits. One realm in which these tools might be valuable is in the upcoming US elections, and indeed, for the forthcoming elections around the world where search results may be less stable and more prone to manipulation [17].

Another critical realm in which these tools can be useful is in building upon qualitative findings on the associations that search engines assign to different groups [19] and the unique queries that they formulate [24]. However, we assert that successful endeavors in these research veins will depend on interdisciplinary collaborations, mixed-methods approaches, and engagement with the existing literature that already exists for each vein.

Limitations

It is worth noting several limitations of our method. First, it does not capture any specific individual's web search experience. For each search query conducted, our method provides a snapshot of the current relationship between the search engine, the structure and content of the web, and all users who searched the same query. Second, it currently only collects the first page of search results. However, people frequently do not browse past the first page of search results, and if these data are required for a specific research project, our package could easily be extended. Lastly, our method does not capture content that loads dynamically, such as the content in drop down menus. One way to solve this issue is to add a selenium-based approach and simulate a browser to interact with and capture dynamically loading content.

Ethics

While not being able to capture a specific user's experience is a limitation in terms of ecological validity, it is an advantage in terms of ethics. The data we collect removes the human element of putting personally identifiable information (PII) in search queries [14], and the PII that renders on a SERP when someone logged in conducts a search. The primary advantage that this serves is with respect to open science: there are no privacy constraints in releasing data collected using this method.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and others for invaluable comments on this project.

REFERENCES

[1] Andrea Ballatore. 2015. Google Chemtrails: A Methodology to Analyze Topic Representation in Search Engine Results. *First Monday* 20, 7 (June 2015). https://doi.org/10.5210/fm.v20i7.5597

[2] Marsha Berzon and Clifford Wallace. 2019. hiQ Labs, Inc. v. LinkedIn Corp.

[3] Munmun De Choudhury, Meredith Ringel Morris, and Ryen W. White. 2014. Seeking and Sharing Health Information Online: Comparing Search Engines and Social Media. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14*. ACM Press, Toronto, Ontario, Canada, 1365–1376. https://doi.org/10.1145/2556288.2557214

[4] Nicholas Diakopoulos, Daniel Trielli, Jennifer Stark, and Sean Mussenden. 2018. I Vote For— How Search Informs Our Choice of Candidate. In *Digital Dominance: The Power of Google, Amazon, Facebook, and Apple*, Martin Moore and Damian Tambini (Eds.). Oxford University Press, New York, NY.

[5] DuckDuckGo. 2018. Measuring the "Filter Bubble": How Google Is Influencing What You Click. https://spreadprivacy.com/google-filter-bubble-study/.

[6] The Economist. 2019. Google's Algorithm – Seek and You Shall Find. *The Economist* (June 2019), 81 (US). https://twitter.com/J_CD_T/status/1137063752606605314 https://twitter.com/RERobertson/status/1137746023131074560.

[7] Robert Epstein and Ronald E. Robertson. 2015. The Search Engine Manipulation Effect (SEME) and Its Possible Impact on the Outcomes of Elections. *Proceedings of the National Academy of Sciences* 112, 33 (Aug. 2015), E4512–E4521. https://doi.org/10.1073/pnas.1419828112

[8] Robert Epstein, Ronald E. Robertson, David Lazer, and Christo Wilson. 2017. Suppressing the Search Engine Manipulation Effect (SEME). *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (Dec. 2017), 1–22. https://doi.org/10.1145/3134677

[9] Susan Gerhart. 2004. Do Web Search Engines Suppress Controversy? *First Monday* 9, 1 (Jan. 2004). https://doi.org/10.5210/fm.v9i1.1111

[10] Michael Golebiewski and danah boyd. 2019. *Data Voids: Where Missing Data Can Easily Be Exploited*. Technical Report. Data & Society. 51 pages.

[11] Laura A. Granka. 2010. The Politics of Search: A Decade Retrospective. *The Information Society* 26, 5 (Sept. 2010), 364–374. https://doi.org/10.1080/01972243.2010.511560

[12] Lucas D. Introna and Helen Nissenbaum. 2000. Shaping the Web: Why the Politics of Search Engines Matters. *The Information Society* 16, 3 (July 2000), 169–185. https://doi.org/10.1080/01972240050133634

[13] Chloe Kliman-Silver, Aniko Hannak, David Lazer, Christo Wilson, and Alan Mislove. 2015. Location, Location, Location: The Impact of Geolocation on Web Search Personalization. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC '15*. ACM Press, Tokyo, Japan, 121–127. https://doi.org/10.1145/2815675.2815714

[14] Declan McCullagh. 2006. AOL's Disturbing Glimpse into Users' Lives. https://www.cnet.com/news/aols-disturbing-glimpse-into-users-lives/.

[15] Connor McMahon, Isaac Johnson, and Brent Hecht. 2017. The Substantial iInterdependence of Wikipedia and Google: A Case Study on the Relationship between Peer Production Communities and Information Technologies. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2018)*. AAAI, 10.

[16] Danaë Metaxa, Joon Sung Park, James A. Landay, and Jeff Hancock. 2019. Search Media and Elections: A Longitudinal Investigation of Political Search Results. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (Nov. 2019), 1–17. https://doi.org/10.1145/3359231

[17] P Takis Metaxas and Yada Pruksachatkun. 2017. Manipulation of Search Engine Results during the 2016 US Congressional Elections. In *Proceedings of the ICIW 2017*. 6.

[18] Eni Mustafaraj. 2018. Are Google's Top Stories Politically Biased? It's Complicated. https://medium.com/@enimust/are-googles-top-stories-politically-biased-it-s-complicated-e9c68e269ed9.

[19] Safiya Umoja Noble. 2018. *Algorithms of Oppression: How Search Engines Reinforce Racism*. New York University Press, New York.

[20] Ronald E. Robertson, Shan Jiang, Kenneth Joseph, Lisa Friedland, David Lazer, and Christo Wilson. 2018. Auditing Partisan Audience Bias within Google Search. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (Nov. 2018), 1–22. https://doi.org/10.1145/3274417

[21] Ronald E. Robertson, David Lazer, and Christo Wilson. 2018. Auditing the Personalization and Composition of Politically-Related Search Engine Results Pages. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. ACM Press, Lyon, France, 955–965. https://doi.org/10.1145/3178876.3186143

[22] Piotr Sapiezynski, Wesley Zeng, Ronald E Robertson, Alan Mislove, and Christo Wilson. 2019. Quantifying the Impact of User Attention on Fair Group Representation in Ranked Lists. In *Companion Proceedings of The 2019 World Wide Web Conference on - WWW '19*. ACM Press, San Francisco, USA, 553–562. https://doi.org/10.1145/3308560.3317595

[23] Daniel Trielli and Nicholas Diakopoulos. 2019. Search as News Curator: The Role of Google in Shaping Attention to News Information. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, Glasgow, Scotland Uk, 1–15. https://doi.org/10.1145/3290605.3300683

[24] Francesca Tripodi. 2018. *Searching for Alternative Facts*. Technical Report. Data&Society. 64 pages.

[25] Nicholas Vincent, Issac Johnson, Patrick Sheehan, and Brent Hecht. 2019. Measuring the Importance of User-Generated Content to Search Engines. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 13.