Matthew Tralka

GEOG 498G

Final Project Paper

The solidification of digital interconnectedness brought about by advanced technical developments and the omnipresent modern smartphone have exponentially increased geographic data presence and throughput in the online sphere (Lazer and Radford 2017; Arribas-Bel 2014). This shift towards online and interconnected systems has vastly expanded the human digital footprint - the world generates 2 exabytes of geographic data every day (VoPham et al. 2018). Several initiatives such as OpenStreetMap and Wikimapia aim to leverage this newfound near-ubiquitous online audience to generate actionable geographic information from voluntary user participation (Goodchild 2007). Such efforts of widespread geographic data collection are known as Volunteered Geographic Information, or VGI. However, not all efforts of VGI are explicitly voluntary. User interaction with the modern smartphone and associated breadth of attached sensors generates a constant stream of information around the web (Senaratne et al. 2017). Traditional sources such as social media sites of Twitter, Flickr, or Instagram record and integrate these datapoints into their products. Many of these social media sites then offer this data alongside semi-public application programing interfaces, or APIs, which are selectively searchable by the technical public (Arribas-Bel 2014). In these cases, companies aggregate broad data voluntarily from users. However, the users may not inherently expect their data to be distributed or used in such a way (Scassa 2013). This form of VGI, subsequently referred to as Involuntarily Volunteered Geographic Information, or iVGI, is the predominate format of VGI that I will discuss.

Data Access & Inequality

As previously established, iVGI is inherently generated by all human interaction in the online modern world. Conversely however, access to the aggregated collections of iVGI is not equal throughout the human populous. Even when companies promote public access to iVGI through APIs, access is immanently limited to users who have the technical ability and resources to decode and interface with such a system (Lazer and Radford 2017). For that reason, computer science related professionals encompass much of the research carried out on these big datasets. Data collection and preparation is often the most resource-intense task associated with such big-data systems (Dasu 2003). Considering the extensive educational, financial, and computational resources needed to interact and process usable data from such a system, accessing iVGI datasets becomes an issue of personal resources and socioeconomic standing, thus having substantial implications in global equity. Furthermore, considering the powerful, positive, and real-world implications that iVGI often carries, the existence of such equitable access issues perpetuates worldwide socioeconomic divisions and injustices.

In the following technical review, I describe the process used to integrate and visualize public APIs on an online website for use by the non-technical public with the goal of alleviating equity issues in iVGI dataset access. It is my hope that providing increased access to iVGI datasets to global parties of interest will help to answer geographic questions around the world and boost awareness for data privacy and equity. For convenience, I named this equitable access website as GeoQuery.

Development Ideals

To ensure equitable non-biased access to iVGI, I pre-established several ideals to guide my development and technical implementation of the GeoQuery platform. The core principle of GeoQuery is openness in data, code, and function. To this end, I licensed GeoQuery under GNU's General Public License Version 3 (GPL V3). Under GPL V3 all interested parties may use, modify, and distribute GeoQuery for any function with no allowance for sublicensing (Smith 2014). This open-source approach

to software licensing ensures that the codebase will be entirely and freely accessible to all interested users no matter their background or resource capacity. From here, I then established that the site form and function itself must be as permissible and accessible as the license which constrains them. Specifically, this means that the site must abide by known accessibility standards from all web-enabled devices with no loss of usability or function. In addition, GeoQuery is only as accessible as its codebase. Thus, I implemented GeoQuery in full, modern compliance with code-quality standards such as PEP8 (Rossum et al. 2008). Combined, this permissible open-source license and commitment to frontend and backend accessibility sets the stage for firmly addressing the problem of iVGI data access equity.

Backend Implementation

In deployment, GeoQuery's backend revolves around a dependency-light stack of Flask (uWSGI), Redis, Nginx, and PostgreSQL. The predominant language of implementation is in Python, a language known for its beginner friendliness and flexibility (PSF 2020). I constructed this specific technical stack for its powerful yet accessible interface, potential for scalability, and open-source nature. I strengthened the overall web-stack by leveraging several relevant packages that fulfill key webserver functionalities.

The first of which is webserver security and user authentication, conducted through the Python packages of Wezkzeug Security, Flask-Login, and WTForms. These services work in tandem across both the backend and frontend systems to provide comprehensive user authentication, input sanitation, and password salting/hashing. Overall, to prevent malicious abuse of GeoQuery systems and resources, I require that all users register with accounts in the GeoQuery database. I enforce this using Flask-Login. Any user who attempts to search without login authentication is redirected internally to a login or sign-up page. On either page, I use WTForms to render and validate all user input to ensure compliance with data requirements. In addition to validation, WTForms provides protection against a popular form of site vulnerability, cross-site request forgery (CSRF). In the case of new signups, once a user provides valid

data inputs, their password is sent encrypted to the backend system where their password is salted and hashed before entry into the database.

Another import webserver task is asynchronous task queuing and messaging. For this, I implemented Celery, an asynchronous task queue mechanism, and Redis, a task que message broker. Implementing these enables the successful completion, management, and communication of long-timed tasks - in this case, API requests- that, if executed in the main Flask task, would hold-up the main web server process and prevent other users from successfully interacting with the server. Celery integrates with Redis for message brokering between celery tasks and the main Flask process. When users successfully authenticate a search request, their search parameters are passed to a Celery task which is communicated to, and executed by, registered Celery Workers through Redis. From there, the task parameters are entered into the database system.

The database system used in GeoQuery consists of two main parts, PostgreSQL and SQL-Alchemy. I chose PostgreSQL for deployment due to its open-source nature, powerful feature set, and overall flexibility. If required, developers can swap PostgreSQL out one-to-one for any other database such as MySQL, MariaDB, or SQLite depending on user requirements and preferences. This is possible due to GeoQuery's use of SQL-Alchemy, an open-source object relational mapper (ORM). SQL-Alchemy simplifies interaction with the database models, creating an improved code experience which benefits code-readability and modifiability.

The final core element in the GeoQuery tech stack is Nginx and uWSGI. Nginx operates as the primary webserver for directing HTTP(S) and static content within and outside the GeoQuery deployment environment. uWSGI is an application manager that replaces the built-in Flask development server in deployment, thus increasing performance and security. uWSGI is responsible for serving dynamic content and the main Flask application in compliance with the Nginx webserver.

Frontend Implementation

I constructed GeoQuery's frontend (Figure 1), or user accessible environment, using semantic HTML, CSS, and vanilla JavaScript. This straightforward implementation assures sufficient and modern performance across varying levels of client compute platforms. To ensure visual continuity of the client experience, I leveraged the popular CSS utility framework, Tailwind CSS. In summary, Tailwind CSS provides a class-based interface to modern CSS standards implemented directly into the HTML markup. This both simplifies the development experience and creates a uniformly consistent codebase for easy open-source maintainability. In addition to accessible CSS, I implemented JavaScript with only the minimal required dependencies for successful GeoQuery operations. In contrast to popular JS frameworks, this implementation method is temporally consistent and straightforward.

I expect most users will interact with GeoQuery through the online interface. Therefore, visual accessibility of the frontend environment is where accessibility matters the most. To this end, I constructed the frontend with full regard to accessibility as defined by the Web Contact Accessibility Guidelines (WC3 2018). To start off, I selected colors that provide sufficient contrast (3:1 contrast ratio) between all points of background and foreground. This ensures consistent readability and usability for users who may be color or sight impaired. I further identified interactive and navigational elements through additional means such as substantial transition animation changes, correlated color groupings, and predictable element placement. Combined with semantic HTML tagging, GeoQuery's initial design operates as a sufficient but improvable steppingstone to full web-accessible compliance.

API Implementation

During the GeoQuery conceptualization process, I identified four main social media APIs - Flickr, Twiter, FourSquare, and Facebook - which contain relevant geographic information to GeoQuery's

function and allow for non-commercial fair-use. For GeoQuery's initial implementation and proof of concept, I decided to solely leverage the Flickr API.

Flickr provides free registered RESTful API access up to 3,600 hits per hour for non-commercial use (Flickr 2020). The Flickr API contains numerous methods but the most pertinent to GeoQuery is flickr.photos.search which permits geospatially constricted searches across five billion Flickr photos. To wrap this API method, I implemented an object-oriented Python class leveraging the Python packages of requests, pandas, and geopandas. Notably, this implementation of the Flickr API method is usable outside the GeoQuery platform, thus facilitating an off-line low-code way to access Flickr geospatial data (Figure 2). For full functionality, this avenue requires slight additional modifications to the currently implemented codebase. Within the GeoQuery platform, I conducted extensive internal testing to ensure the wrapper class provides fast and fair access to the Flickr API. Currently, I have validated GeoQuery's Flickr API wrapper to ~600,000 results in ~30 minutes with a program average of 1.5 minutes per 20,000 photos.

Due to GeoQuery's class-based architecture, further implementation of the before-mentioned APIs is straightforward. The functionality and design of GeoQuery is highly modular and API method calls take place separately from core GeoQuery webserver functions. Therefore, future contributors can easily modify and adapt the GeoQuery platform to implement personalized APIs to fit their specific usage scenarios. This further adds to the overall flexibility and accessibility that the GeoQuery platform provides to the geospatial community.

Deployment

For ease of deployment, I implemented the above frontend and backend systems through the containerization software, Docker. Docker containers, akin to the traditional 'virtual machine', are compartmentalized self-sufficient environments known for their ease of deployment and platform

compatibility (Docker 2020). Specifically within the Docker environment, GeoQuery leverages Docker-Compose, a tool for orchestrating multi-container Docker applications. Using Docker-Compose, interested parties can deploy the complete GeoQuery platform in one line of code (Figure 3). This has substantial implications for users of all backgrounds who want to implement their own customized GeoQuery system.

User-Deliverables

Once deployed, the GeoQuery platform is easily accessible – and searchable - through the GeoQuery web-interface. Once a user successfully completes a search, the GeoQuery system returns two main items of interest, an interactive map visualization through Leaflet.js and downloadable files containing search results. Within the map visualization, there are three layers of search-point representations: heat, cluster, and absolute (Figure 4;5;6). Each layer brings a unique form of qualitative representation to the overtly quantitative datasets returned by GeoQuery. The heatmap layer is particularly useful for areas with concentrated values, the cluster interface is exceptionally suitable for areas with overlapping points, and the absolute layer provides a simple true-to-dataset visualization. Pivoting from there, GeoQuery presents the user with two file formats to download the collected data in: Geographic JavaScript Object Notation (GeoJSON) and Comma Separated Values (CSV). Combined, both downloadable formats flexibly address the expected file-format needs of GeoQuery's technical and non-technical end users alike.

Data Use in Literature

In popular literature, the data provided by GeoQuery – known as VGI – is used for a broad range of research due to its wide breadth and relative accessibility. Often, VGI is used to supplement the temporal and spatial scale of traditional authoritarian datasets. The openness and flexibility of VGI contrasts starkly with the conditions, fees, and exclusivity needed for dataset partnerships with

companies and institutions (Arribas-Bel 2014). The implications of VGI carry substantial weight; VGI has been used to supplement research in human mobility, geospatial artificial intelligence, and environmental fields (VoPham et. al 2018). However, VGI is not without compromise. Due to inherent spatial and temporal biases in data collection, VGI often provides a modified perception of true-to-life patterns. In particular, VGI carries substantial urban biases that unevenly represent overall human patterns (Hecht and Stephens 2014). However, due to VGIs decentralized nature, these implicit biases differ than ones found in traditional datasets. Thus, incentivizing a joint-dataset approach. Another concern in VGI research is data privacy and legality which may compromise the effective legal use of certain data sources (Scassa 2013). These legalities are often unclear, therefore possibly jeopardizing individual intellectual property and overall research efforts.

One example of iVGI use in literature is Hale and Tralka (*in review*). The authors used a pre-release version of GeoQuery to collect geotagged photos from Flickr to assess the biodiversity of the botanical landscape in northern Ticino, Switzerland between 2015 and 2020. Specifically, the authors collected photos taken in and above the subalpine zone (defined as 1500 masl), collecting up to ten photos per user. Then, the authors completed a meta-content analysis to identify plants in the photos, collect tags and comments related to plants, and assess the overall user sentiment towards conservation, botany, and the environment. Initial assessment shows that while visitors in the subalpine zones of Switzerland appreciate the flora of mountain areas, they rarely recognize critical species or environments. Furthermore, by leveraging the spatial element of these photos and plant identifications as proxies for environmental awareness, the authors were able to identify significant below-average awareness areas. Thus, allowing the optimal distribution of environmental awareness initiatives and resources. This geovisual analysis, similar to the suggested cross-disciplinary efforts in Andrienko et al. (2007), showcase the possible wide-reaching use scenarios of iVGI. Without GeoQuery, replicating this methodology falls outside the realm of non-technical researchers and activists.

Next Steps

Overall, GeoQuery is in an operational and effective state. However, there are several “next steps” needed to ensure continual success. I will continue the implementation of further APIs to increase the informational breadth of GeoQuery. Limitations around API implementations revolve around continued API access. This reliance is of concern and I am examining alternative routes to collect data. Regarding data processing, I would like to improve the analytics side of GeoQuery. Perhaps implementing several statistical elements (EX. Local Moran’s i) on the search results page. Next, I would like to flesh out the codebase documentation and frontend design to ensure successful communication of GeoQuery’s ideals and function. Finally, once I am acceptably satisfied with system stability, deployability, and accessibility I plan to “advertise” GeoQuery to ensure the program can be found by interested users and contributors.

Figures

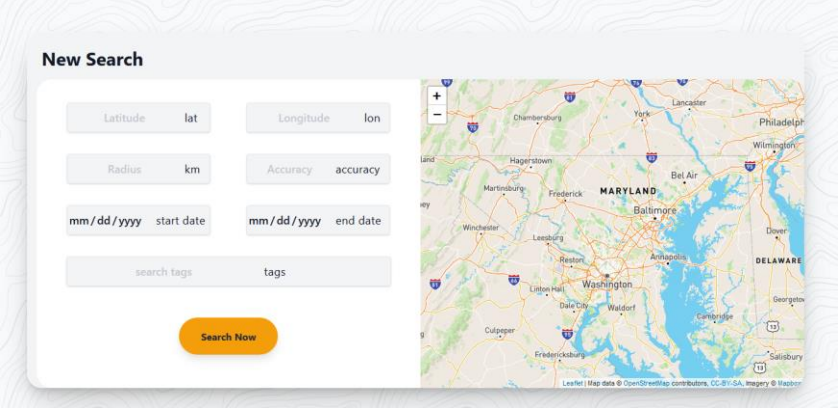

Figure 1.

Frontend example of GeoQuery's "New Search" interface.

Figure 2.

Mock example of GeoQuery's Flickr API call.

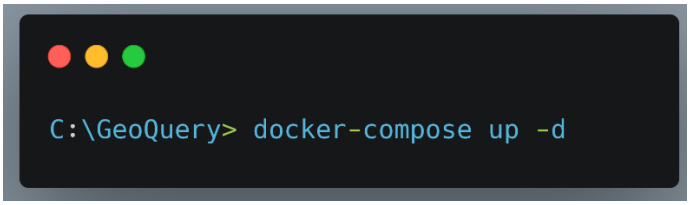

Figure 3.

Mock example of the one-line required for GeoQuery deployment

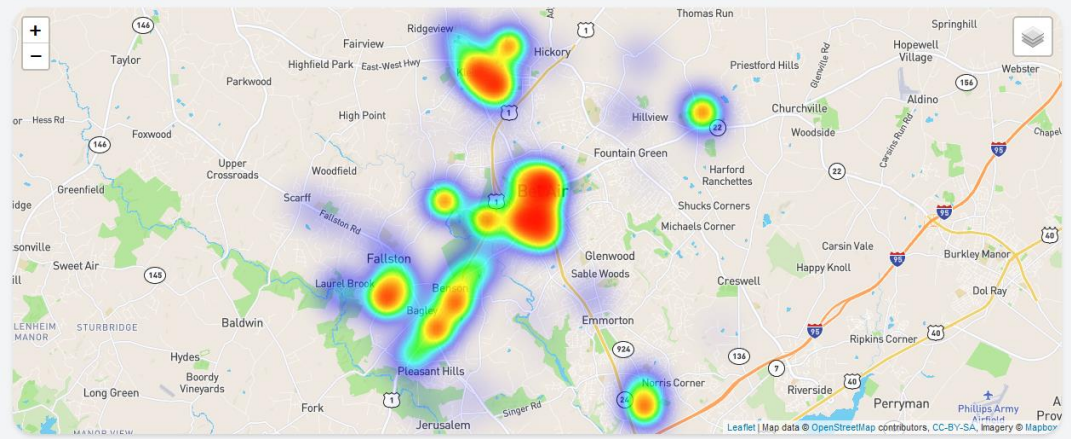

Figure 4.

Example of GeoQuery's "heat" results layer.

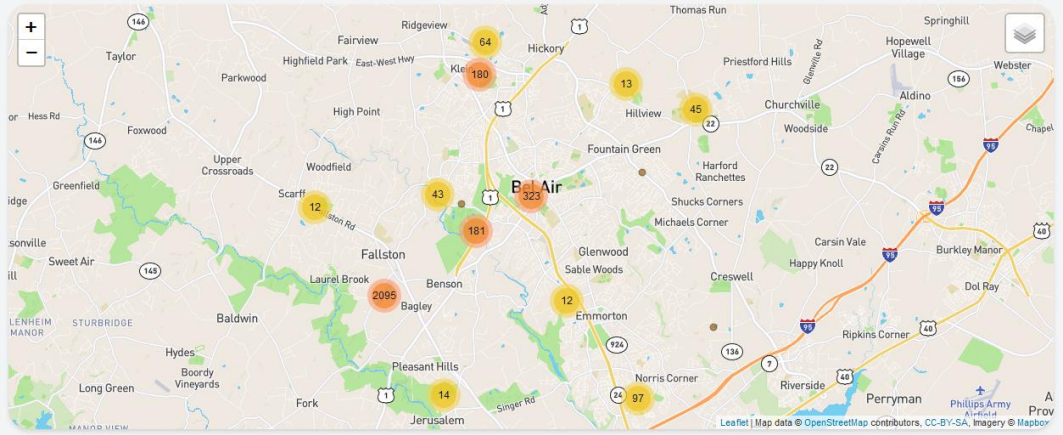

Figure 5.

Example of GeoQuery's "clustered" results layer

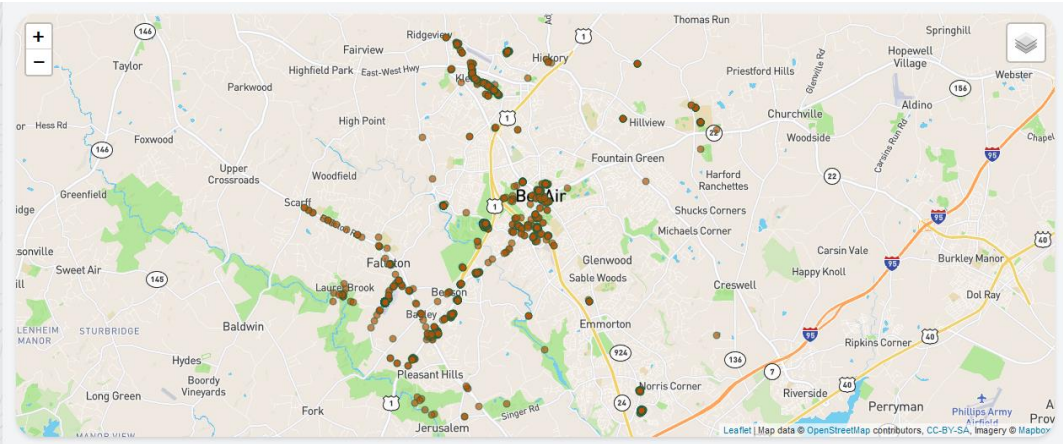

Figure 6.

Example of GeoQuery's "absolute" results layer.

References

Andrienko, G., N. Andrienko, P. Jankowski, D. Keim, M.-J. Kraak, A. MacEachren, and S. Wrobel. 2007. "Geovisual Analytics for Spatial Decision Support: Setting the Research Agenda." *International Journal of Geographical Information Science* 21 (8): 839–57. https://doi.org/10.1080/13658810701349011.

Arribas-Bel, Daniel. 2014. "Accidental, Open and Everywhere: Emerging Data Sources for the Understanding of Cities." *Applied Geography* 49 (May): 45–53. https://doi.org/10.1016/j.apgeog.2013.09.012.

Dasu, T, and T Johnson. 2003. "Exploratory Data Mining and Data Cleaning."

Docker. 2020. "What Is a Container?" Docker.Com. 2020. https://www.docker.com/resources/what-container.

Donoho, David. 2017. "50 Years of Data Science." *Journal of Computational and Graphical Statistics* 26 (4): 745–66. https://doi.org/10.1080/10618600.2017.1384734.

Flickr. 2020. "Flickr API Services." Flickr.Com. 2020. https://www.flickr.com/services/api/.

Goodchild, Michael F. 2007. "CITIZENS AS SENSORS: THE WORLD OF VOLUNTEERED GEOGRAPHY," 15.

Hecht, Brent, and Monica Stephens. 2014. "A Tale of Cities: Urban Biases in Volunteered Geographic Information." *Proceedings of the 8th International Conference on Weblogs and Social Media*, 197–205.

Lazer, David, and Jason Radford. 2017. "Data Ex Machina: Introduction to Big Data." *Annual Review of Sociology* 43 (1): 19–39. https://doi.org/10.1146/annurev-soc-060116-053457.

[PSF] Python Software Foundation. 2020. "About Python." Python.Org. 2020. https://www.python.org/.

Rossum, Guido van, Barry Warsaw, and Nick Coghlan. 2001. "PEP 8 -- Style Guide for Python Code." Python.Org. 2001. https://www.python.org/dev/peps/pep-0008/.

Scassa, Teresa. 2013. "Legal Issues with Volunteered Geographic Information: Legal Issues with Volunteered Geographic Information." *The Canadian Geographer / Le Géographe Canadien* 57 (1): 1–10. https://doi.org/10.1111/j.1541-0064.2012.00444.x.

Senaratne, Hansi, Amin Mobasheri, Ahmed Loai Ali, Cristina Capineri, and Mordechai (Muki) Haklay. 2017. "A Review of Volunteered Geographic Information Quality Assessment Methods." *International Journal of Geographical Information Science* 31 (1): 139–67. https://doi.org/10.1080/13658816.2016.1189556.

Smith, Brett. 2014. "A Quick Guide to GPL V3." GNU Operating System. 2014. https://www.gnu.org/licenses/quick-guide-gplv3.html.

VoPham, Trang, Jaime E. Hart, Francine Laden, and Yao-Yi Chiang. 2018. "Emerging Trends in Geospatial Artificial Intelligence (GeoAI): Potential Applications for Environmental Epidemiology." *Environmental Health* 17 (1): 40. https://doi.org/10.1186/s12940-018-0386-x.

[W3C] World Wide Web Consortium. 2018. "Web Content Accessibility Guidelines 2.1." 2018. https://www.w3.org/TR/WCAG/.