Virginia Courts Project

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Matthew Bacon
Spring, 2021

Technical Project Team Members
David Stern
David Alves
Jessie Shen
Andrew Kim

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature Matthew Bacon Date 5/16/2021
Matthew Bacon

Approved Jack W. Davidson Date 13-MAY-2021
Jack Davidson, Department of Computer Science

Virginia Courts Project

Motivation

Currently there is no easily accessed database that contains all of the U.S. court records desired by researchers. Instead, what exists is a patchwork of separate court databases from across the country. For example, in Virginia each district and circuit court as well as the Supreme Court all have separate databases. In addition, the various databases were not set up to look at data in aggregate, but instead designed to find individual court cases. The problem is further exacerbated by each state having their own laws governing which records are made public in their own database systems. Due to this patchwork it is extremely difficult to get a national dataset for researchers to work with. Without this data it is hard to get a clear overall picture on how justice is administered in Virginia and throughout the entire country. For example, researchers at UVA's Law school want to study judicial outcomes / criminal sentencing to determine what inequities exist within Virginia's criminal justice system. For the reasons above, it was decided that there was a need for an easily accessed database where researchers could obtain all the court records they want at once.

Project Description

Web Scraper

In order populate our database with actual court records we need web scraping. Web scraping is the general process of running a program to take and record information from a webpage. Last year, students working on this project wrote a web scraper for General District courts in Virginia. This scraper was written in NodeJS with the puppeteer node package for the web scraping. The scraper would loop through all the general district courts in Virginia on dates listed in a file and extract court cases and put them in a different json file. There were, however, a few problems with this approach. The resulting JSON files with the court records were massive

and not easily navigable with each court record on the same line as well. Therefore, if you tried to view it in a text editor it would be extremely difficult and sometimes crash. In addition, it was impossible to select a specific court to scrape, or to just scrape civil or criminal court cases. Lastly, when scraping through several courts we realized that certain data was missing and not being scraped off the website. Data was missing when a court case had multiple hearings, defendants, or plaintiffs.

To address these issues with the old scraper several things were done. First, the massive JSON court record files were broken up by year and court. More specifically, for every court that was scraped a folder was created containing folders for each year. Inside these yearly folders there were JSON files, one for criminal cases, and one for civil cases. Additionally, the contents of the JSON files were reorganized with multiple line breaks in-between court records for easier navigation.

Next, I added command line arguments to allow one to select which court to scrape, year they want to scrape and whether they want to scrape criminal or civil cases. Command line arguments are additional commands written after the command to start the scraper when calling it in a console as show below.

Figure 1.

These command line arguments give greater control to the user to pinpoint what exactly which court records to scrape without having to scrape all the courts. This is especially beneficial if you want to break up the job of scraping courts given that this process is extremely time consuming. For example, we could scrape civil cases on one computer and criminal cases on another to theoretically double the rate of scraping.

Additionally, to make a scraper easier to run, I created a bash script. This bash script would let them more easily interact with the scraper. For example, one could see which commands they can run on the scraper or which courts they could scrape. Another option that the script allowed was to run multiple scrapers in parallel, or concurrently, to allow for faster scraping of court records. Slowness in the scraper is an issue unfortunately, because the website that we are scraping from blocks users who are going through court records too quickly.

Figure 2. Bash Script

Lastly, I fixed the bugs in the web scraper that caused certain fields to be missing when there were multiples of them (i.e. multiple plaintiffs).

In addition, to getting the Virginia General District Court scraper in working order, I also added the ability to scrape Virginia Circuit Courts to the scraper. This took the largest amount of effort because the circuit court records were on a completely different website than the general district courts. Therefore, most of the previously written code was incompatible. To build the

scraper, I used a similar system to the general district courts. I used puppeteer to click on various buttons and fill in forms to navigate through the website. Then when the navigation led to a page with pertinent information it would scrape it and put it into the relevant JSON file.

There are a few things that made it easier to scrape circuit court records than general district court. The website containing the circuit court records does not contain all the mechanisms that the general district court website had to prevent automated systems from scraping data. First, the circuit court website did not contain recaptchas. These are the buttons you have to click to verify that you are not a robot. These captchas would produce pictures and our scraper would have to click on the squares containing certain objects as shown below.

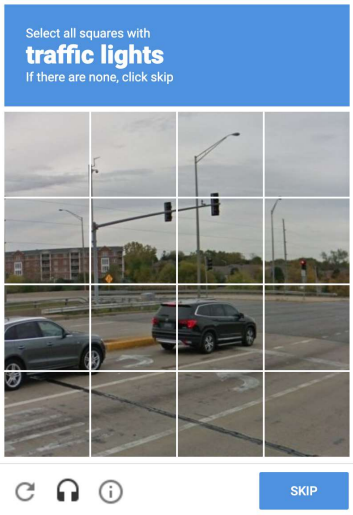

Figure 3. A Recaptcha

In order to get around these in the general district courts, we had to use a third-party API that cost money that would enable our scraper to click on the correct squares. Although, this solution worked it slowed down the scraping process. Secondly, the circuit court website allowed us to navigate as quickly as possible. This was not case previously, where we would have to wait at least 700 ms between pages in order to prevent being halted by the website from navigating further. Luckily, these were not issues when scraping the circuit court.

Additional work on the web scraper has also been done. I wrote some unit tests for the web scraper using the java script test framework Mocha. These tests are written using code and when run, call functions in the web scraper and then see if the output of these function is correct. The tests help ensure that the web scraper runs as intended. This is useful when making changes in the code to make sure nothing was broken. The tests written, test the extracting of data from district court cases by making sure all the information needed was collected, and test if the date command line argument is functions as intended. In the future, more tests could be written to verify other parts of the web scraper are functioning. Another improvement that could be made is to implement changes to our shared git repository to prevent any changes that cause a test to fail.

In order to effectively and collaboratively work on the web scraper our team used GitHub. This is an online repository where one can make changes to a codebase locally and then push said changes to a shared repository in the cloud where others can then download, use, and make their own changes. The aforementioned web scraper and changes made to it can be found at https://github.com/SCOUTAPP-DB/virginia-scraper.

Database and Website

After we scrape court records, we need a way to share them with others. To do this, last year students created a website, using the web framework Django, that allowed others to download court records. However, there were several issues with this website. The first issue was that the data format to upload data was different than the format of the scraped data. Therefore, we could not easily update the scraper with now scraped information. Secondly, the database was not set up in an efficient way. Therefore, we ultimately decided to create a new database as well as a program to upload and anonymize court records collected from the scraper.

When creating a new database teammate Jessie Shen created new database schemas based on what data we want to collect. Database schemas are just a description of what data a

database will contain and the relationships between certain data. They were created in a way to minimize the amount of storage taken up. We needed to four different schemas for the following different types of court records, civil circuit, civil district, criminal circuit, and criminal district. The following is one of the schemas created.

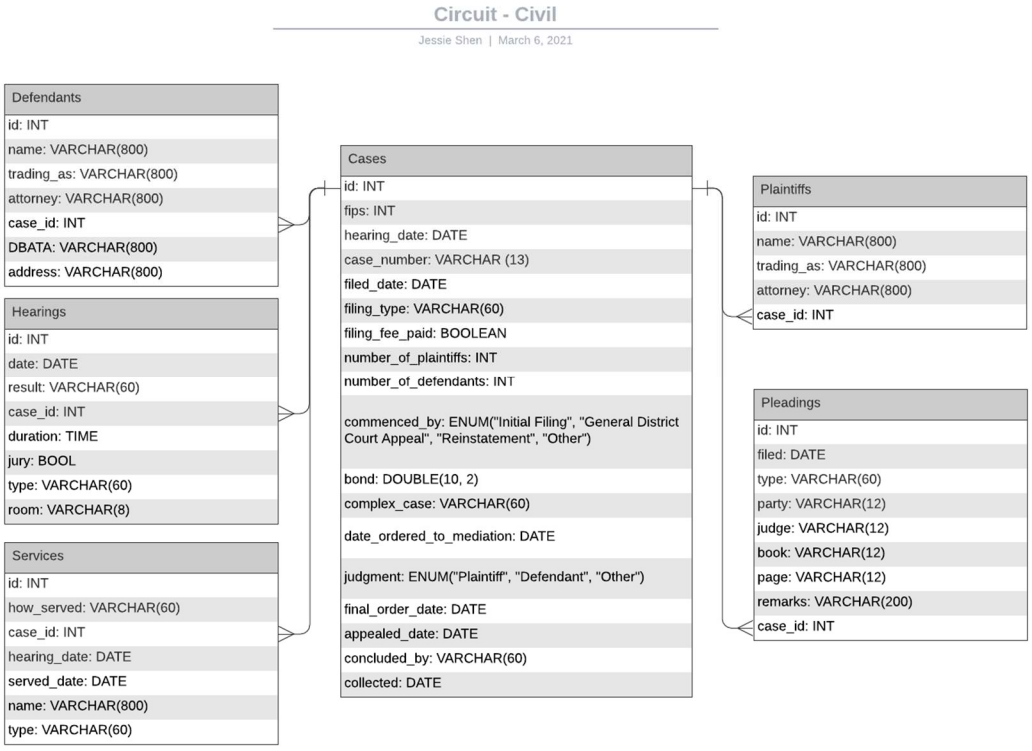

Figure 4. Schema for Civil Court Records from Virginia Circuit Courts

Each of the blocks in the diagram represent a table in the database, and all the items in table are names of the data that it will have along with the type of data it will contain. The databases will also be made using SQL or Structured Query Language. This is a language to communicate with a database. It allows for easy handling of structured data.

With the Database design in place, we still need a way to connect scraped data to the database itself. Group member David Stern worked on writing a program to do this. This program will read the JSON files generated from the scraper and generate SQL commands to insert the data into the database. In addition, team member David Alves and Andrew Kim

worked on making a program to allow users to more easily interact with the database and be able to retrieve data from it. The eventual goal is that this program will be integrated into the website to allow for users to easily access scraped data.

Future Work

Although progress is made, more work needs to be done for this project. First, we need to create the databases specified in the schemas Jessie Shen created. This can either be done using a cloud computing platform like Amazon Web Services (AWS), or could probably also be done using University of Virginia's computing resources. AWS would require additional cost and cause some loss of control. However, AWS is highly reliable and provides easy to use services to build a server using Docker which previously was being used for the database in this project. In addition, this database would then have to be populated. Currently the programs to upload the JSON files to the database are incomplete and need to be worked on. This program also needs to anonymize court records. The current plan to do this is to hash the names of the defendants and plaintiffs, or in other words generate an encoded token based on the name. This way researchers using our data could tell which court records pertain to the same individuals without easily know their identity. In addition, the program to access the database also needs to be incorporated into the website. Lastly, the UI of the website needs to be improved.

Once we have all the components of the website and database fully integrated and complete, we will need to fill it with scraped data. Currently, we only have data for a year on a couple small courts. Therefore, we will need to scrape years' worth of data using the scraper. This will take an extended period of time. When the scraper was run on Albemarle District Court it took around 24 hours to completely scrape just one year. So, when scraping all of Virginia, it will likely take over a week to scrape one year. There are a few ways to speed this up however.

For example, you could run multiple computers to scrape different years or courts to speed up the process. The only concern is that you do not want to run the scraper on too many machines at once and overwhelm the Virginia State Court websites that we are scraping. However, once this is done, we should only need to re-scrape the websites periodically to obtain new court records.

Potentially even more can be done on this project. Virginia is not the only state with online court records, and collecting information from other states could be beneficial to researchers. In future years additional states could be added, and the scope of this project could be expanded. This will involve creating new scrapers, new database schemas and programs to upload the data. However, with enough time and human capital this can be done.

Conclusion

Over the course of this year, our team has furthered the development of the Virginia Courts project. With the improved functionality of the web scraper, we can now scrape all the information we need from both Virginia District Courts and Virginia Circuit Courts. In addition, we now have a solid plan to fully implement our database and website. Hopefully, we can have a fully integrated and working system soon. This would allow researchers to easily access bulk court records data in Virginia. This data is extremely beneficial for searching for inequities in the criminal justice system in Virginia. Despite the additional work needed, we have made great strives in developing the Virginia Courts Project.