Big Data Design and Analytics with COVID-19 Using Jupyter and MongoDB

Nicholas Pierce

Indiana University Purdue University Indianapolis

1. Introduction

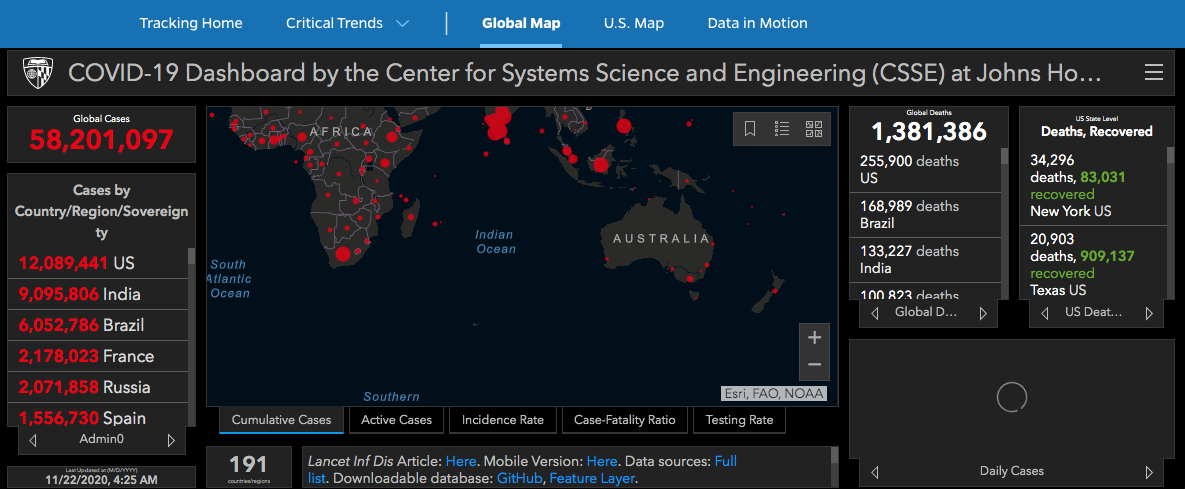
This semester we studied and evaluated the nature of an ever-expanding collection of data, driven by cheap computing and ever more complex analytical tools. The study of large amounts of data -dubbed *Big Data-* is one of the subfields of databases that has exploded over the past several years. The characteristics of said data are complex, large, exponentially scaling over time, and typically challenging to deal with, given quality control and definition challenges.

There are numerous methodologies that one can take to address a Big Data project, each with benefits and challenges. One could heavily rely on database design, having strong relational/NoSQL providing a system that is highly efficient, scalable, and able to handle large amounts of data with high access time and accuracy. One could also focus on combing through large amounts of existing data, focusing on combining different kinds of data into a multitude of different databases under a data warehouse solution.

One of the key aspects of a Big Data project is providing information and to drive predictive analytics that is useful and contributing to business or the community. It is simple to simply download a JSON[[1]](#footnote-1) file, upload it to a cluster, and query data, but if it lacks consistency or structure the results are useless. Therefore, I decided to look at the current challenges and opportunities collecting and analyzing data around the COVID-19 epidemic. I not only explore the large swathes of data available with the COVID-19 epidemic, but built a fully functioning educational Jupyter Notebook that walks the reader through the steps of building a Big Data application, including data wrangling, sanitization, querying, and displaying. This area lacks development; actually, learning Big Data tools, querying, and visualizing is not an easy task, as the class experienced this semester, as a large portion of the Big Data implementation is stuck in research papers and advanced studies in data engineering.

1. Related work

In terms of related work, there are two main sections that could be the focus of a Big Data project depending on the outcome that is desired. If data insights and analytics are the focus, then there are many more comprehensive COVID-19 dashboards and models than time would allow during this final project. As an example, the Johns Hopkins Coronavirus Resource Center COVID-19 dashboard shown in Figure 1 is highly robust, interactive, and customizable. My focus for this project was not to make a comprehensive, COVID-19 analysis modelling application, as a quick Google search will yield hundreds if not thousands of dashboards for every city and country imaginable.



*Figure 1: Johns Hopkins COVID-19 Dashboard*

The other major area of related work is the educational result. From my extensive research, there were few resources available to simply learn how to walk through the entire process of building a Big Data project from inception to display. There are data science/ engineering blog posts that deal with components or small subsections of the design and implementation, but almost none of them ever bridged the gap between disparate information resources. It is incredibly difficult to find solid educational resources for Big Data applications.

1. Approach

When I began looking at COVID-19 data, it is organized mostly as a country-centered daily record, spread across multiple CSV files with key information: country name, states/ regions, confirmed cases, deaths, latitude and longitude, etc. The data, unfortunately, was inconsistently named and missing key members, making any analysis challenging, even though the data sets only encompass a year or so of information. Therefore, the first step I had to take before ever deciding on a database scheme was to first perform data cleansing.

My focus was to utilize the Python library *pandas[[2]](#footnote-2)* to manipulate the data as a collection of data frames instead of loading and editing the CSV files directly. There are many powerful functions that I used as a part of the pandas library allowing the dropping non-unique duplicate data, dropping all not a number data, and other data inconsistencies. This allowed me to get essentially one large CSV file as a DataFrame, which allows the greatest flexibility for data integration.

My first approach was to construct a simple relational database out of the combined CSV files with each of the key elements about the country as column headers (Table 1). I quickly realized that combining the data into one SQL table was not going to be efficient at all for building or querying. This led me to a JSON solution.

*Table 1: Sample .csv file of a single day of COVID-19 data*

After looking at several sources online searching for a more simplified solution, I consistently saw JSON as being the most popular format with the greatest flexibility. Essentially, each of the countries is a single object within the list of objects in the file, where each element contains a list of the country information as stated above. The typical scheme of each country document can be seen in Figure 2. The datatypes for each of the fields are fairly standard, with either large integer, integer, string, date, etc. One of the things I did discover, though, was the “coordinates” datatype which was incredibly helpful in plotting the locations of large spikes in cases later on.



*Figure 2: Country Document as JSON Object*

With a large document of each country as a JSON object and an easy way to store all of the countries, I decided to use a *document store database scheme* with this JSON file as the center. Reflecting on our semester, I remembered how easy and convenient it was to use MongoDB to query large documents, so MongoDB Atlas was my choice for a database store and querying language.

Opening up the database in Atlas shows a bit more about the intricacies of the database design and some of the decisions that were made. First thing to notice is the size of the database, it is incredibly large, the statistics collection which is the aggregate of daily date going back to 2019 has around 270,000 documents and is almost 100 mb in size.

Graphical user interface, text, application, email

Description automatically generated

*Figure 3: Database in MongoDB Atlas*

We can also see that the collection uses a few indexes. On building my database, I followed several tutorials and methods on the purpose and implementation of indexes. As we discussed throughout the semester, MongoDB indexes are a way to bypass doing a linear scan of the entire database. These indexes focus on getting key data the fastest, like largest countries are flagged, highest cases, etc. As the database is incredibly large, these indexes help speed up queries immensely.

With a way to store my data and query it efficiently, the final step was to select a frontend framework to wrap everything into. At my internship with *Eli Lilly and Company*, I utilized Jupyter extensively as a data science and data engineering tool. A Jupyter Notebook was the perfect way to not only utilize the data from my queries to show the results of COVID-19 trending throughout the world, but also expose all of the code in the live code cells and the process so that people can do their own Big Data projects.

1. Results

From a data perspective, I had everything parsed, filtered, and merged to a JSON file from my *pandas* DataFrame. Though as I was learning about setting up my MongoDB cluster, I realized there is a public MongoDB atlas cluster (*http://covid-19.hip2i.mongodb.net/covid19*) with updated information daily from the John Hopkins data I was using, which validated the process I used to sanitize, design, and set up the database scheme. The process was enlightening, and so I included in my Jupyter Notebook the process of building a document store database for people to glean from, while using the public cluster for my actual querying.

MongoDB’s robust functionality is demonstrated with its data collection ability. The PyMongo library sets up a seamless MongoDB connection, allowing for complex queries of several countries documents to gain insights into the shear number of cases. Interestingly as well, we can use MongoDB to gain insights into the actual biases of the data as well. I learned quickly that most countries lack subdivision of data into smaller areas like the United States subdivides into states and localities. Subsequently, the geospatial data is not inherently as accurate as the data combines all cases in some locals under a single latitude and longitude, unlike the US which has each state assigned to its corresponding number of cases.

I perform several of these large queries and use the data as a platform for my UI elements, the Jupyter widgets. These are simple to use widget interfaces that are highly abstracted but have a large amount of functionality. For example, to display visualizations for data created in a COVID-19 dashboard, a user would have to hardcode many elements or utilize JavaScript frameworks or other languages to get visualize the data. In the case of Jupyter, simply install the appropriate package to the environment, and use the corresponding query results converted to DataFrames to populate the widgets.

As seen in the Jupyter notebook, I am able to create dynamic, fully populated tables that are searchable and editable, dynamic maps with charted COVID-19 data, as well as several different forms of graphs that give solid visualizations of country data. As these widgets are fully modular and not custom, they can be adapted to any type of data or data store, as the combination of MongoDB and Jupyter is incredibly powerful and straightforward.

1. Contribution and Innovation

While there are many more advanced and robust COVID-19 dashboards providing insights into the pandemic data, I think the major contribution and innovation from this project comes from the visual and interactive outline of the entire process of building a full Big Data project from scratch. Too many times do we learn about skills in isolation, whether it be database design and data engineering, or building web apps to interface with existing data. Few provide how the various pieces integrate together to educate and inform other engineers on how to build applications for their needs.

The way that Jupyter lays out its code cells and intertwines it with explainable markdown cells lends itself to an incredible learning environment. A student can follow along step by step and understand the reasoning behind decisions and the driving forces of the application development, instead of simply copying code like is very common in the industry. And by the extension of widgets and interactive graphs, Jupyter is by nature directly results driven and encourages information sharing. I strongly believe that the combination of (1) a robust, easy to use database design scheme and querying language of MongoDB via python, (2) the interactive and highly learnable nature of Jupyter notebooks, and (3) the document visualization aspects of Jupyter widgets, allows for this project to be an incredibly useful contribution to the world of Big Data applications.

1. *JSON-JavaScript Object Notation, a standardized, simple format for storing and exchanging data, that is easy to create and easy for machines to parse* [↑](#footnote-ref-1)
2. Pandas-data analysis library written in Python [↑](#footnote-ref-2)