Stefan Doehler

Miles Saffran

CollegeRank Project Report

1. Working Split

Stefan Doehler created the web scrapers, MySQL database connection and design, and data manipulation functions.

Miles Saffran created the Flask web framework for the user interface.

Both worked on developing the MySQL queries.

1. System Functions

Our system supports 3 major functions: data scraping, data manipulation, and MySQL search queries.

1. Data Scraping: Utilizing the BeautifulSoup library, CollegeRank is able to scrape data from 5 different websites (US News, College Raptor, Best Schools, Best Colleges, and Niche), parsing the html from each site and pulling the name, rank, and location of each school listed. These three pieces of information are added to a cumulative list which is then passed to data manipulation functions.
2. Data Manipulation: The goal of the data manipulation functions is to turn multiple data sets into one large data set, containing information that can then be inserted directly into the database. Location is scraped from each website in the form of (city, ST) where ST is the abbreviated name of a state. A function turns these two pieces of information and creates (city, state, region) where state is fully spelled out and region is either West, Midwest, South, or Northeast. The same school may be pulled from two websites with different names (University of California at Los Angeles vs. University of California--Los Angeles or Washington and Lee University vs. Washington & Lee University) so names are also parsed against one another to check for this.
3. MySQL Search Queries: CollegeRank allows queries on all schools in the cumulative list regardless of location as well as queries on both region and state. A user can also set an upper bound on how many schools they wish to return on the interface. As an example, a user can specify that they wish to see the top 10 schools in California or perhaps the top 25 schools in the Southern region.

Together these functions allow CollegeRank to provide a robust, college ranking system. All schools are returned in order based on their average rank when taking into consideration their ranks on each of the 5 websites.

1. System Limitations

The data in our system depends on the data found on our chosen college ranking websites. This data is scraped based off the html structure of each web page, which means that if the html structure were to change, the data could no longer be scraped until the parsing rules were updated. In terms of database functionality, our system cannot query schools based on name.

1. Main Issues and Problems

There were a few challenges that we faced along the development of our project. The first was scraping the college ranking data from the websites. In order to do this, the html from each website had to be studied by hand, so that we could correctly parse the information and not return text unrelated to a college’s rank, name, or location. Additionally, we developed on different operating systems (one with Windows and one with Mac) which led to some compatibility issues. One such problem was created a database server with MySQL that both us could access. In the end we decided to just host the database on a local server. Lastly, integrating the Flask web framework with our MySQL database proved to be a challenging task. It took us a lot of trial and error to correctly display our query information in a structured html table.

1. Main Contributions

After searching around the web for a little, we could not find another college ranking aggregator that worked similar to ours. CollegeRank provides a central, cumulative ranking system to display the best “average” rank. This is very helpful due to the fact that there are many dozens and dozens of ranking systems and it can be difficult to pick one ranking system over another. CollegeRank erases this issue by accumulating an average rank over 5 of the most popular ranking sites. Additionally, being able to query schools based on region or state is a nice added feature that allows a user to examine the best schools in a particular location.

1. Selling Points

Our database contains data that is purely from the real world. In other words, we did not inject any data points ourselves. We take 5 different data sets and mesh them together to create 1 final data set, a feat that required a lot of data scraping and manipulation. Our database uses 3 distinct tables, which are queried over using inner joins. This in itself is not a magnificent feat, but our data is stored very conveniently and efficiently. The following displays our database design.

\*count represents the amount of times a school appeared across the 5 websites

Schools

|  |  |  |
| --- | --- | --- |
| **name**  **(primary key)** | **score**  **(foreign key to score\_ID)** | **location**  **(foreign key to location\_ID)** |
| Yale University | 1 | 1 |
| University of San Francisco | 2 | 2 |

Locations

|  |  |  |  |
| --- | --- | --- | --- |
| **location\_ID (primary key)** | **city** | **state** | **region** |
| 1 | New Haven | Connecticut | Northeast |
| 2 | San Francisco | California | West |

Scores

|  |  |  |
| --- | --- | --- |
| **score\_ID**  **(primary key)** | **average\_rank** | **count\*** |
| 1 | 4.3 | 5 |
| 2 | 27.2 | 2 |