CIS 2250

Team Ottawa

Data Design for Crime Explorer

Introduction

This document describes some of the design choices we made when it came to handling data in our application. The challenge we faced was to develop an effective functional design for our program by analysing several aspects of the data of interest, such as its structure, format, size and availability.

Data Management

Due to the extremely large size of the dataset (over 2.8 million records), reading the data file every time the data is to be used is a very inefficient and slow approach. Therefore, our design choice was to read the entire dataset into memory at program launch, so that it is available for fast access in RAM for the rest of the program's runtime. The weakness of this design lies in being compatible across different systems. A dataset this large would also require a large amount of RAM (at least over 1GB) to be parsed into memory, which limits the functionality of the program on older systems or VM based systems with a low dedicated amount of memory.

Early in the design stages, our program would run for one iteration and then exit. With what turned out to be a 1-minute loading stage, this proved ineffective. The size of the data forced us to design a more efficient program that would load the dataset once and reuse it until the user decides to quit.

In terms of the features of the data, the fact that it was open and freely available from a trusted government agency (Statistics Canada) made it easy to manipulate and manage. The data came in a fairly raw form, but it was verifiable and accessible.

The data was structured in a very systematic manner which allowed us to use efficient search algorithms to retrieve particular records in the dataset. In fact, to answer the question of "Crime Per Capita", our team originally believed it would be necessary to divide the number of violations by the population of each city from a separate, census CSV file. After further exploring the dataset, we were pleased to discover that the "Rate per 100,000 population" statistic provided exactly what we needed without having to consult outside sources.

It would be straightforward to maintain the program and add support for future years of data. With regards to latency, the data appears to change yearly as Statistics Canada updates the information, and its timeliness is around 1.5 years before present day. Currently, our program has the final year of 2015 hardcoded into it. If *Crime Explorer* were to be extended to long-term support, adding each year of data as it becomes available, a dynamically chosen end-date would perhaps be more appropriate.

Data Analysis

Analysis of the data showed that it was encoded with CP1252 formatting and needed to be decoded into UTF:8 before being parsed into memory. Our design needed to accomplish this without modifying the original datafile; therefore, each record in the dataset had to be individually read into memory and decoded before being parsed.

Viewing the datafile revealed missing values for more than a few records. Our program needed to adapt to this situation by displaying appropriate output to the user for missing data. Checking is in place so that missing data is bypassed, and if no data at all is available for a particular statistics, a meaningful message to the user is displayed.

Our final design differed from one of our early deliverables in that we originally planned on using the Coordinate field of the CSV to search for records. However, since *Crime Explorer* features a search engine for user input, we instead found it more effective to separately analyze the GEO, VIOLATIONS, and STATISTICS fields with regular expressions. Perl's built-in string matching functions (*eq*, *ne*) also proved useful in finding matching sets of data.

Correct matches are stored in arrays until the user chooses to output them to a graph. These arrays are reused with each iteration of the program; this way, the memory footprint of the program remains manageable and does not grow in size indefinitely.

Since the program only requires the original dataset and performs parsing in-memory, *Crime Explorer* is portable and can be moved easily to another system.

Data Visualization

Our design choice for data visualization in the program was to implement two main methods of plotting the data.

- For a specific crime in a specific location over a range of years (line plot)
- For a specific crime in a specific year over multiple locations (bar plot)

The plotting went through a few iterations before our team was satisfied with the results; during the final design stages we enabled automatic X-axis scaling and raw value printing overlaid on each bar to enhance readability.

In addition, before sending the data to R for visualization, *Crime Explorer* shows the relevant data within the terminal in text form. This way, the program is more flexible; for example, a stripped-down version could be made for systems that do not have access to R by removing a few lines of code.