Worcester Polytechnic Insitute

A Major Qualifying Project submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfilment of the requirements for the Degree of Bachelor of Science on

March 17, 2016

MATTERS. Data Mining

Bogatov, Dmytro dbogatov@wpi.edu

Hennessy, Jillian jrhennessy@wpi.edu

supervised by

Professor Elke Rudensteiner

Abstract

ABSTRACT HERE!!111

Executive Summary

IT GOES HERE!!111

Acknowledgements

HERE!!111

Contents

List of Figures

List of Tables

Introduction

With the total amount of data in the world growing steadily at a fast rate, the need to analyze large amounts of data in datasets, or big data, is a major key to productivity growth, and innovation [10]. With this increase in information and the detail of the information available across the web, the internet will continue to provide a basis for the continued growth of data in the future. Data is a part of every industry and business in todays world, from retail to government. Big data and analysis can help to create value in these industries by making information more usable, showing more accurate performance measures, and providing detailed analytics that lead to better decision making.

In this data-driven time, it is important to be able to make decisions quickly using the data that is available. Big data visualization is an effective way to present the important information amongst the large amount of data used and helps to drive complex analysis [11]. With the use of big data analysis, it can make otherwise too large amounts of data meaningful to those who looking to interpret significance from it.

The Massachusetts High Technology Council, or MHTC, is a group of technological, professional and higher education executives across the state of Massachusetts. They are comprised of higher education CEOs, senior executives from Massachusetts and more. For over thirty-eight years, MHTC has advocated for various policies and programs to create and keep both a healthy and competitive business climate. Their goal is to establish and keep Massachusetts as a competitive place for successful business and talent building, including a specific focus within the technology sectors. In part because of the MHTC and its members, who are regarded as the regions most venerable technology association, Massachusetts is regarded as one of the top competitive areas for businesses that are high tech [12].

MATTERS, the Massachusetts Technology, Talent, and Economic Reporting System, was developed by

MHTC along with Worcester Polytechnic Institute (WPI) and other institutions as a collaborative effort designed to aid users in understanding a variety of public data in an effort to help make Massachusetts the top location for high technology businesses. MATTERS uses this data collected to both measure and evaluate the current state of Massachusetts compared to other states in the country and provides policy makers and advocates with easily accessible and searchable dynamic data to assist in their efforts regarding decisions to help retain and grow business in the state. MATTERS is able to provide a way to centralize a wide range of talent, cost, and economic metrics and state rankings from federal and state government sources, non-profit organizations and media outlets into one location for its users [13].

Previously to our project, WPI IQP, MQP and graduate student teams worked to develop the MATTERS dashboard as it was when we began working. The goal of our Major Qualify Project (MQP) is to provide additional features to the MATTERS website that will give our users additional ways to interact with the data and metrics found within the site. The idea for the metric builder was presented by the MHTC members as a means for the MHTC to be able to visualize their own indicator that they were in the process of creating. Users will be able to create an account on the MATTERS site and be able to build and display the results of their own unique metric formulas for ranking states. Users will also be able to retrieve any of the data points found throughout the site for their own personal use through the creation of our own API. These new features will provide authorized users an opportunity to use and share the data in even more innovative ways, aiding in the MHTCs goal toward making Massachusetts a leading state for high technology business. These two features create further access to the sites data and additional ways to manipulate and visualize the data to interpret it in new ways. These features also create an expanded user base for MATTERS and different privileges for different types of users, such as the already existing administrative users, and general users who also would like to create their own indicators.

Background and Related Work

The creation of a metric builder page and an API for the MATTERS site is a result of two prior project groups and additional efforts by WPI students who worked to establish the initial features and the design framework for MATTERS to exist. The initial development of MATTERS came from a team of students completing their Interactive Qualifying Project (IQP) with the aid of other WPI graduate student teams. This team of IQP students researched and came to the resulting decisions regarding the front-end visualization for the MATTERS dashboard [1]. This includes the inclusion of decisions regarding colors, and visualization types such as charts and tables, that would be used to the represent the data within the system in a manner that was both intuitive for the users and a quick way to show analytics. In parallel, the graduate student teams worked to develop the back-end for MATTERS. They made it possible to extract the data wanted from various sites and then parse and clean this retrieved data, before then uploading it to a database. This data would be called upon to be shown in the various visual methods for the users [2].

Following this initial front-end and back-end design and implementation of the MATTERS dashboard, a team of students working on their Major Qualifying Project (MQP) for WPI then worked to improve the administration center and the data integration for the dashboard. This involved the data integration pipeline manager as well as the Administration Centers development to allow for MHTC and other administrators who had access to be able to easily upload their desired data and metrics, as well as an easy way to view this data on the administrative end. This MQP team was able to integrate 20 new data sources for the MATTERS dashboard, make the Administration Center useable for non-technical users, and improve upon the look and feel of the MATTERS dashboard as a whole [2].

2.1 System Architecture

2.2 Requirements

The main goal of creating an API for MATTERS as well as creating the Metric Buidler page for the site and what it would entail, required looking into other similar features of different sites and making decisions regarding implementation, design and security features for these aspects of the MATTERS site.

2.2.1 API and Documentation

An API is an Application Programming Interface. APIs are tools that allow other programs to interact with the software program that the API belongs to without having to give someone complete access to your codebase. In a sense they expose some of the internal workings of software to the public but in a way that is limited by what the API chooses to provide. APIs aid in providing a way to share a potentially large amount of data in an efficient way. According to the chief data officer of Philadelphia, Mark Headd, APIs allow a specific audience to use data more quickly, easily, and efficiently when they are looking to do something specific with the information. APIs are beneficial for not only keeping certain aspects of code private, but also for saving users time. Even in cases of open source programs where all of the code for a program is visible, the number of code can be so large that it is inefficient to search through the whole codebase for specific data [3].

APIs will be beneficial from both a business aspect and for programs that aim share information and research with many people. An advantage of an API being an automated tool is that it can process a large number requests without any added cost and work from the developers. APIs also present the data requested in a manner that is useful and easily manipulated for the users purposes. For APIs whose goals are to share certain data across systems there is a great benefit efficiency wise, as many manual procedures are cut out by the APIs automatic generation. When these tasks are done manually, they waste time and are both laborious and repetitive, as well as being more prone to error; all of these things being costly on the developer end. While it is more time consuming and costly to develop an API initially, the overall benefit once it is developed is very clear [3].

Additionally, APIs not only encourage innovation and the ability to manipulate and extrapolate data by external collaborators, they provide a means to do so that is more secure for the group creating the API. They reduce the risk of how and what data is obtains and encourage good practice for how to properly manage the data available [4].

One important step to gaining the full benefits from an API is making sure that the API is secure. This can be achieved through requiring authentication and authorization of users before they gain access to the APIs features. It is good practice to ensure that the users are who they say they are and that they are given permission before using data or parts of a developers program. This also allows for a record of who exactly is accessing the data, in the event that there is an issue so that a specific user can be tracked down [5].

In addition to creating the API for a site, it is necessary to create good documentation to accompany the API, to make the API as user friendly and simplistic as possible. API documentation provides developers with everything that they need to know in a concise manner in order to use the APIs features with other applications. The API documentation needs to contain the classes, functions and other important aspects of the API [6]. The documentation guide allows developers to easily interact with a sites API and their own code, as many developers style their code in their own unique way.

2.2.2 Metric Builder

The main feature that is to be developed is the metric builder page which will be available for the registered users on the MATTERS website only. A user will be able to select any number of the metrics already available in the MATTERS database and combine their data to create their own personal formula or indicator. The user can then use their created formula to display the states rankings based on this indicator. This will allow users to see and evaluate states based on what they think is most important as a whole or combination of various factors, and interpret the data in their own way, simply. By allowing users to create their own data set from all the metrics available and then provide their own weights to each of these data values, we are allowing users to see the value that each state has based on how important they find a large number of metrics.

It is important to ensure that many different data types are able to be combined to create one final value. It is also important to consider the effects that certain data types will have on the final value received

from the users indicator. For example, the MATTERS database contains rankings, percentages, nominal values, and other data types. Certain data types could end up dominating in a user created indicator when combined with significantly smaller value data types. There are also complexities involved with certain data types that are represented in the reverse of the majority of the data types. For example, both rankings and some other data types use numerical values in which a lower value indicates a better performance by the state, versus the majority of other metrics in which the larger the value is, the better a state is performing in that area.

Finally, there are many metrics that do not contain a value for every single year that other metrics contain. Situations where one metric being used in the users metric builder equation does not contain values for years that another metric being used in the same equation need to be considered. There needs to be a uniform solution for what to do in these instances in a way that does not deceive the users of the indicator, as this would affect the calculations and following visualizations across years where these issues exist.

These factors need to be taken into consideration in order to provide accurate and understandable visualizations and results when users create their own indicator in the metric builder. It is for this reason good to consider special handling or removal of certain data types, as well as potential ways to normalize the final numerical values received from the formulas the users have created in the metric builder.

We looked at various economic indicators to come up with an idea for how to allow our users to combine sets of data to use for comparisons. One common approach in economic indicators is to use weighted averages [7]. For example, one of the most popular economic indicators, CPI, or the Consumer Price Index, uses weighted averages of data from hundreds of different consumer goods and services to measure inflation and deflation. Each of the goods considered by the CPI is weighted based on its importance [8].

Weighted averages work by taking each number in a given data set, and multiplying the value by its given weight. This will give you a new value reflecting the product of the data and its weight. Add all of these products together from the data set to get the total value. Then, add up the total of all the weights. Finally divide the total value by the total weight in order to receive the weighted average of your data set. The weights are used to show the importance that each specific item in a data set has in order to calculate an overall value of all the data together [9].

Methodology

3.1 Design Choices

3.1.1 Metric Builder

Implementing a Metric Builder the core question was how to store and compute a user metric. We have come to these two options

1. Store a user metric data in the databse
    - Pros
        - Fast to retrive; no computation needed
    - Cons
        - Waste of space in the database
        - User metric data remains ststic; update in the underlying data does not cause an update in the user metric
2. Store a user metric as a metadata only and compute values on-the-fly (on-demand)
    - Pros
        - Efficient use of the database space
        - User metric data is always in sync with underlying data
    - Cons

- Might be time-consuming to compute

Looking at the cons of each option, we attempted to approximate how bad they are. Having n users each creating m user metrics each of which has data for k years we have $m \cdot n \cdot k \cdot 50$ entries in the database. This data would be a snapshot. If any data point in the metrics changes, user metrics remain out of sync.

For the second option, it was diffucult to approximate time it takes to compute a user metric as it depends on the server load and the number of metrics involved in the user metric. We implemented the second option and benchmarked it wth 3 users trying to compute largest possible user metric (all metrics with all states selected). It took around 1.5 seconds to load all 3 user metrics. This drove our design decision towards the second option - computing metrics on-the-fly.

3.1.2 API

Implementing an API, there are 2 problems to consider

- Do we implement a closed API or open API? If closed one, how do we implement authentication.
- Which methods do we put in the API?
- How our query string will look like? What is the output format?

Authentication

Although MATTERS system is a public resource, we wanted to at least track who uses our API, how and how often. This requirement made us chose a closed API option. Thinking of authentication we had a few options

- Bind to user
- Simple *API key* (token) authentication
- Complex *handshake* mechanism
- *Handshake* mechanism with random value

Binding to API user t regular user would allow any registered user to access API. Although this kind of authentication is the easiest one to implement we wanted a separate API users.

Simple *API key* authentication is the one when API users are given unique token (long random string) which they need to provide with each request. Based on this token system is able to accept or deny connection and trach individual users.

Handshake mechanism with random value involves the following steps

- Client sends in username (not password)
- Server responds by sending back unique random number
- Client encrypts using user's password plus number as key
- Client sends hashed or encrypted value back to server
- Server encrypts using the user's same password plus number as key
- Server compares two hashed or encrypted values, if same then grant access

Provided that the main purpose of the API being closed is tracking an activity for statistical purposes, *API key* aouthentication was impplemented.

Methods

Main method implemented in the API is `api/data?`. This method returns all data points for given metrics, states and years.

Since `api/data?` needs specific IDs as arguments, there is a need for supporting methods which show information about metrics and states. So we also implementd `api/metrics?` and `api/states?`.

Query string and output format

The are two options for constructing a query string.

- `api/method?argument1=value1&argument2=value2&...`

- `api/method/value1/value2/...`

To make out API user friendly we implemented the first option since it clearly shows which value corresponds to which arguments and these argument-value pairs may be in any order.

The final query strings are given is Listing. 3.1

Listing 3.1: API methods examples

```
1  /* method */
2  /api/data?metric=&state=&year=&apiKey=
3  /* example */
4  /api/data?metric={16,32}&state=MA&year=*&apiKey=key
5
6  /* method */
7  /api/metrics?apiKey=
8  /* example */
9  /api/metrics?apiKey=key
10
11 /* method */
12 /api/states?apiKey=
13 /* example */
14 /api/states?apiKey=key
```

The output format JSON (JavaScript Object Notation) was chosen as it both comapact and human and computer readable.

3.2 Changes to the database

We had to add two tables to the database - **APIUser** and **User Metric**. See Table. 3.1 and Table. 3.2.

Name	*Type*	Comment	Reference
Id	*INT*	Unigue aoutoincremented identifier	none
Name	*VARCHAR*	API user's name	none
ApiKey	*VARCHAR(160)*	Unique random string; used for authentication	none

Table 3.1: **API User** database relation

Name	*Type*	Comment	Reference
Id	*INT*	Unigue aoutoincremented identifier	none
Name	*VARCHAR*	User metric's title	none
Value	*TEXT*	JSON formatted list of metric id - coefficient pairs	none
UserID	*INT*	Id of a user - author of the user metric	Id in **User** relation

Table 3.2: **User Metric** database relation

Testing

4.1 User tests for metric builder

4.2 Changes made

Results

5.1 Metric Builder final design

5.2 API final design and documentation

Conclusion

6.1 Recommendations

References

[1] Alex Fortier, Long Hoang Nguyen Duc, Kevin Mee, and Westley Russell. Advancing massachusetts technology, talent and economy reporting system (matters). http://www.wpi.edu/Pubs/E-project/Available/E-project-042915-094249/unrestricted/Advancing_MATTERS.pdf, 2015.