Assigning programming languages to geographic locations based on the activities of GitHub users

Gelera, Rodney
University of Victoria

McCulloch, Kaileen
University of Victoria

Warwick, Nick
University of Victoria

April 4, 2017

Abstract

Data has never before been generated at such high speeds. With the ever increasing volume of data becoming available every second, it has become more and more challenging to find meaningful information. Data mining and data visualization are two tools that can help solve this problem. In this paper we discuss the effectiveness of visualization as a data mining tool. We use a classification technique to simplify information and a visualizer as a quick and easy evaluation method. In order to explore these techniques we use GitHub user data to investigate the use of programming languages according to geographic locations. We used a GitHub API to scrape user programming language and location data. In addition, we used Googles GeoCoding API to convert textual locations to latitudinal and longitudinal coordinates. From that we classified each unique geographic location with a single programming language. We built a visualizer using MapBox GL JS which allows us to explore the geographic distribution of programming languages. From this, we concluded that there are trends in programming language popularity and that visualization is indeed a useful data mining tool.

Contents

1 Introduction **3**

2 Data Collection **3**

3 Data Pre-processing **7**

4 Data Visualization **8**

5 Conclusion **11**

List of Figures

1 Map of all languages 9

2 Map of JavaScript 10

3 Map of PHP 10

1 Introduction

Fundamentally, data mining is an information extraction technique for identifying patterns and trends in large datasets. It is used to derive conclusions from datasets where manual searching is not feasible due to constraints on memory or time or because the raw data is visually incomprehensive. There is a broad range of techniques and with the surge of `big data`, it is becoming even more important. Every digital process generates data. This data can help companies monitor their systems and prevent threats or it can help companies understand customer engagement and adapt to the ever changing ecosystem. GitHub, an online version control platform, is one such system producing endless data just waiting to be mined. It is particularly interesting because the data produced from GitHub allows us to track the changes in the programming. There is a large variety of development platforms available, selecting which tools to use can be a challenge. Whether you are a professional looking to stay relevant in this competitive climate or a person just starting their career in tech, looking at the language of choice in your area will help you make a more informed decision. The focus of this paper will be on mining the distribution of programming languages based on geographic location. This paper will first discuss the data collection approach and limitation, it will then proceed to cover the the data pre-processing methods. From there it will explain the visualization technique used and what conclusions could be drawn from it.

2 Data Collection

For data gathering we used GitHub's own API to collect usernames[1]. There were three possible requets we could make with GitHub's APIi, they are listed below. However, it had a rate limit of 60 requests per hour which we predicted to be a problem. Therefore, we looked for alternative solutions.

- `GET /users?since=0`
  - Returns a list of 30 users with limited information starting from specified id

  -
    ```
    [
    {
    "login": "octocat",
    "id": 1,
    "avatar_url": "https://github.com/images/error/octocat_happy.gif",
    "gravatar_id": "",
    ```

[1]https://developer.github.com/v3/

```
"url": "https://api.github.com/users/octocat",
"html_url": "https://github.com/octocat",
"followers_url":
"https://api.github.com/users/octocat/followers",
"following_url":
"https://api.github.com/users/octocat/following{/other_user}",
"gists_url":
"https://api.github.com/users/octocat/gists{/gist_id}",
"starred_url":
"https://api.github.com/users/octocat/starred{/owner}{/repo}",
"subscriptions_url":
"https://api.github.com/users/octocat/subscriptions",
"organizations_url":
"https://api.github.com/users/octocat/orgs",
"repos_url":
"https://api.github.com/users/octocat/repos",
"events_url":
"https://api.github.com/users/octocat/events{/privacy}",
"received_events_url":
"https://api.github.com/users/octocat/received_events",
"type": "User",
"site_admin": false
},
{
...
},
...
} ]
```

- GET /user
  - Returns a list of repositories of the given user
  - {
    ```
    "login": "octocat",
    "id": 1,
    "avatar_url": "https://github.com/images/error/octocat_happy.gif",
    "gravatar_id": "",
    "url": "https://api.github.com/users/octocat",
    "html_url": "https://github.com/octocat",
    "followers_url":
    "https://api.github.com/users/octocat/followers",
    "following_url":
    "https://api.github.com/users/octocat/following/other_user",
    "gists_url":
    "https://api.github.com/users/octocat/gists/gist_id",
    "starred_url":
    "https://api.github.com/users/octocat/starred/owner/repo",
    "subscriptions_url":
    "https://api.github.com/users/octocat/subscriptions",
    "organizations_url":
    "https://api.github.com/users/octocat/orgs",
    "repos_url":
    "https://api.github.com/users/octocat/repos",
    "events_url":
    "https://api.github.com/users/octocat/events/privacy",
    "received_events_url":
    "https://api.github.com/users/octocat/received_events",
    "type": "User",
    "site_admin": false,
    "name": "monalisa octocat",
    "company": "GitHub",
    ```

```
    "blog": "https://github.com/blog",
    "location": "San Francisco",
    "email": "octocat@github.com",
    "hireable": false,
    "bio": "There once was...",
    "public_repos": 2,
    "public_gists": 1,
    "followers": 20,
    "following": 0,
    "created_at": "2008-01-14T04:33:35Z",
    "updated_at": "2008-01-14T04:33:35Z"
    }
```

- `GET /users/:username/repos`
  - Returns information about the given user

One possible alternative solution was to scrape data directly off the HTML. We found a project on GitHub that did just that[2]. The input for this application takes a GitHub URL such as:

- a user profile (ex. https://github.com/username)
- a user's repository page (ex. https://github.com/username?tab=repositories)

With this scraper, we wouldnt have to worry about the rate limitation when requesting information on GitHub users. However, one problem we had with it was that it was not up-to-date with the latest GitHub UI. Being an open source project, we were able to edit the scraper to return only the information that we needed.

For both solutions, we would still need a list of users. The only way we would be able to get this list of users is the `GET /users` request from the GitHub API. With the rate limit, we were able to get 1800 users per hour with 60 requests. The GitHub API returns some user information with this request, but not the information we needed such as location, thus, the reason we use the scraper to get user information. We did this 13 times to gather a list of 23400 usernames.

[2]https://github.com/nelsonic/github-scraper

3 Data Pre-processing

With a working scraper and a list of usernames, we created a script that uses our modified scraper to get the users location and most used programming language based on their repositories. We skipped users that did not set a location, users that returned a 404 error, and organizations. This gave us a final total of 10241 users, 43.7% of our initial list of users. This may have been due to the fact that not many people set their location on their profile.

GitHub allows users to set their location as a free-form text. This can be problematic for getting exact coordinates for a user's location. Google Maps has a geocoding API which converts text strings of geographic locations into latitudinal and longitudinal coordinates. Unfortunately this API has a usage limitation of 2,500 free requests per day. With a paid API key, we were able to exceed this limit. The downside to using this API is that we will only be able to get coordinates for whatever the user sets as their location. It can be as specific as a street address, it can be as vague as just a country name, or it may not be a real location at all. The Geocoder API tries its best to find the best matching coordinates. For example, if a user sets its location to Canada, the coordinates returned are in the middle of northern Saskatchewan. These results may not be 100% accurate but it gives the general location of where the user is.

After getting the users coordinates, the script finds the users most used language. GitHub classifies a repository as a certain programming language based on how many lines of code that project contains. Our script looks at this attribute for each of the users repositories and finds the most frequent programming language used by the user. In the case of a tie, all the languages that tied are kept. Once we have the users coordinates and the users most used programming language(s), the script adds the user data to a GeoJSON formatted JSON.

The user data needed to be aggregated on a geographic location basis, in order to classify each unique location by a single programming language. This was a multi-step process. Firstly, all users with the same geographic coordinate were merged into a single instance. During the merge a dictionary of programming languages for each location was maintained which tallied the number of users using each language. For users with multiple languages, that user would count once for each one of their used language. The language used by the most users was then chosen as the winning language. If languages were tied for most used, an arbitrary language, other than 'NULL', was selected as the winning language. Obviously, this is a very naive way of approaching this problem. A better method would be to apply a weight to each usage. To demonstrate this consider a simple case of a geographic location which has 'Ruby' and 'JavaScript' tied for most used with one user each. Now say the JavaScript user only used JavaScript in one project and the

Ruby user used Ruby in hundreds of projects. In this case it would make more sense to classify the location as Ruby since it's used in more projects for that area. Applying a usage weighting would do exactly that, whereas, with the arbitrary selection each language has a 50% chance of being selected.

Additionally, the data retrieved in the *Data Collection* step needed to be converted into a GeoJSON format in order to be input into the data visualizer. Conversion from JSON to GeoJSON was a simple matter of adding one additional geometry feature, namely, 'coordinates'.

4 Data Visualization

The data visualization portion of our project involved creating a method for reading our data and displaying it on a map, representing the location of a point such that it matches its respective geographic coordinates. We also had to figure out a method for representing the respective programming language of a given data point. We decided to encode programming languages as colours, since it provided a wide range of possible values to represent the 200 languages we were tracking, in addition, GitHub already represents different programming languages as colours, so it seemed fitting. After researching various methods for doing data visualization in JavaScript, we decided to use Mapbox GL JS, a JavaScript library that renders interactive maps using WebGL (Web Graphics Library)[3]. WebGL js a JavaScript API that allows for the rendering of interactive 2D and 3D graphics in a compatible web browser without the need for plugins[4]. The use of WebGL allows for extremely fast and smooth interactions.

[3]https://www.mapbox.com/mapbox-gl-js/api/

[4]https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

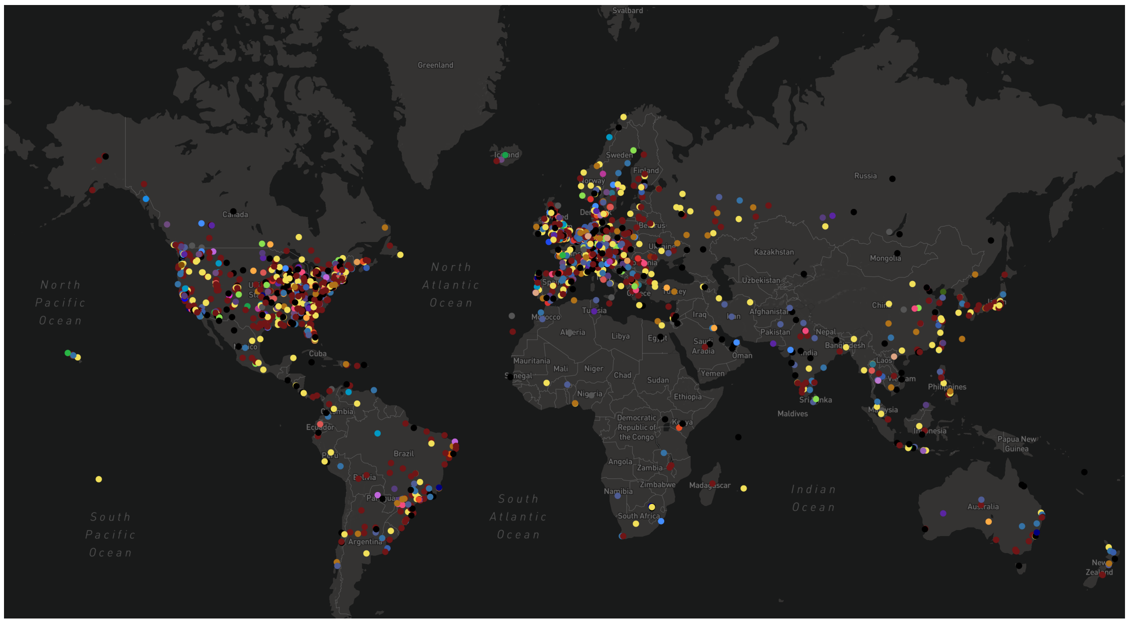

Figure 1: Map of all languages

Figure 1 represents a map with all of our data points shown. At first glance, it is clear that the yellow dots (JavaScript) and the dark red dots (Ruby), are the most prevalent. We see clusters of data points which represent geographic locations with large numbers of GitHub users, these clusters for the most part are in locations where we would expect them, i.e. on the east and west coast of the US, as well as in Europe.

When representing all of the data points on a single map, it is hard to see any correlation between geographic location and popular programming languages. However, when filtering and plotting data points for specific languages, it becomes easier to see potential trends in the data.

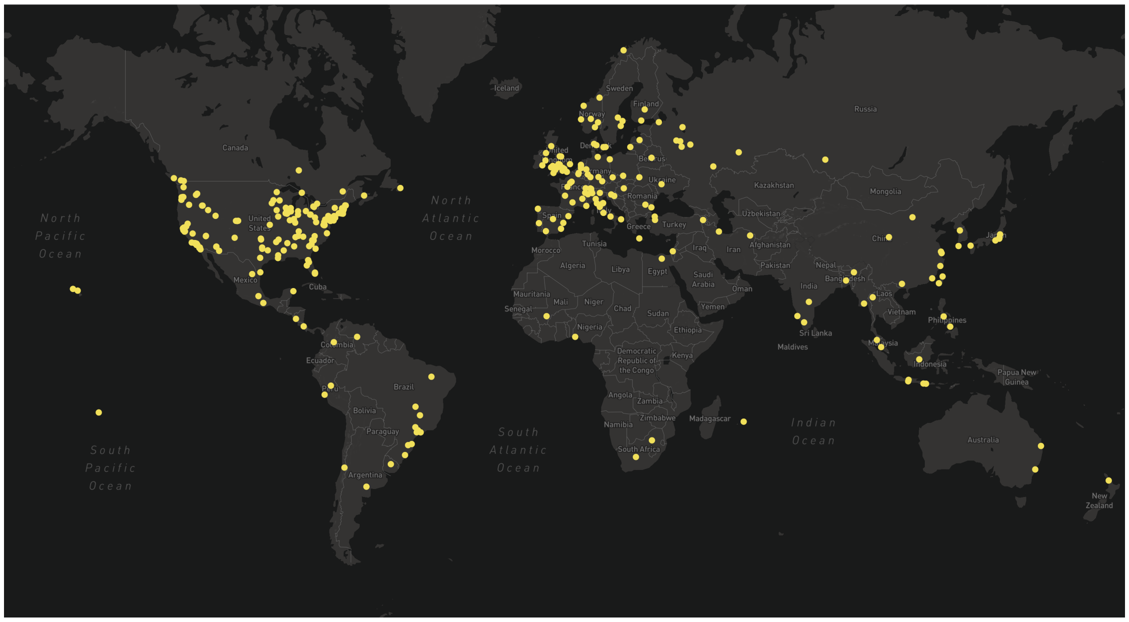

Figure 2: Map of JavaScript

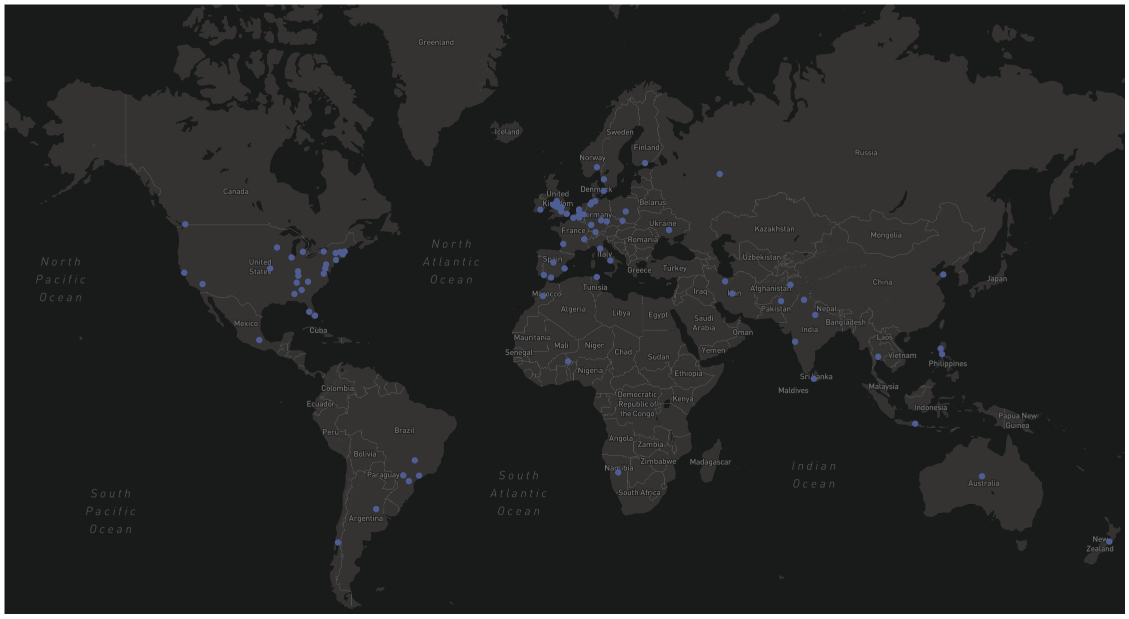

Figure 3: Map of PHP

Figure 2, with the yellow dots, represents the use of JavaScript, whereas Figure 3 represents the use of PHP. As you can see, for the most part the languages have similar clusters, however, JavaScript is much more popular on the west coast of the US, whereas PHP has almost no influence in the west. This could be due to the tech hubs in the west coast favouring JavaScript over PHP.

5 Conclusion

In conclusion, we found that there does seem to be small trends in programming language popularity (for some languages) potentially related to geographic locations. However, we would need to analyse a larger dataset to confirm our findings. From our visualization, we also found that Ruby and JavaScript are the most popular programming languages in North America and Europe, at least according to our dataset. Ultimately, this application helps to show the power of data visualization in extracting information from large and complex datasets.

There are multiple streams of future work for this project. One possibility would be creating a heatmap view rather than plotting points. Another possibility would be adding a feature that allows for a timelapse, showing the different trends in programming language usage over time. Using a larger dataset would also be beneficial in getting a more useful visualization.