DIR: Developer Industry Readiness

Dylan Lomax, John Carmack, Brandon Mingledorff, Lillian Coar

Abstract - With the growing importance of formal education for developers to find employment, an obvious question is whether or not that formal education actually prepares the developer for the workforce. We explore that question by asking if the programming language developers are taught in their programs are actually in demand and are present in their work lives.

I. OBJECTIVE

It is our goal to answer the question of whether or not a developer's first language is indicative of their success in the software engineering workplace. We wish to ascertain
whether or not the programmer's choice of beginning language, or the language that their curriculum is taught in has any impact whatsoever in their career prospects, or in their ability to adapt and assimilate to a working environment typical to the modern software engineer, which seems incredibly saturated with positions requesting experience with web technologies and JavaScript frameworks.

II. DATA USED

A. Industry Skills Data

The data for the industry portion of the project was obtained from a large web scrape of over 41,000 job postings on CareerJet, a job search engine that pulls job data from over 58,000 other websites and serves this data in a developer-friendly way. The purpose of this is to make it as simple as possible to navigate and tabulate this data. It was for these reasons that CareerJet was chosen over other job search forums, as others did not allow the volume of page requests needed or offered an API that did not have the level of functionality necessary to obtain the data needed to meet the project's objectives.

To determine the most sought-after industry skills, a search for 'developer' jobs on CareerJet was done and mentions of skills (with an emphasis on programming languages) in each of the job postings were counted to see which ones appeared the most overall. Two large datasets were obtained to further validate the data, one that included 27,242 different jobs in the 30 most populous U.S. cities, and another with 41,074 jobs searched by each of the 50 U.S. states and Washington, DC. Data for each city and each state was stored in a separate CSV file to see if any patterns could be found in different areas of the country. An overall result was determined by taking every CSV file in a dataset, adding it together, and storing that into a 'main' CSV file for overall results analysis and visualization.

B. Developer Languages Data

The data for the languages that developers use was gathered from the World of Code (WoC) database. WoC contains all publicly available git information on most git repositories available. This allowed us to pull developer data without having to go through the process of gathering this information ourselves.

For the purposes of this project, we needed two sets of data: Developers and the languages that they use. To do this we use WoC to gather a list of all publicly available developers and first and last commit SHA1's associated with that developer. We then mapped the commits to the files associated with them and used their extensions to determine the languages used.

III. MODELS / ALGORITHMS USED

A. Industry Skills Algorithm

To obtain the data from CareerJet, a Python script was written using BeautifulSoup that makes requests to the website using carefully constructed URLs that included certain parameters that ensured the data obtained was relevant. The result of each request was the underlying HTML of a single developer job posting in a particular city or state. This result was parsed by first zeroing in on the relevant content of the web page (the job title and description) and then removing the HTML tags. Each word in the body was then looped through by removing end-of-sentence punctuation using regular expressions and then splitting the words of the body into an array by whitespace. When looping through the array, if the word was present in a predefined list of skills and programming languages, an entry for that word in a 'counter' object was incremented by one. Once every job posting that the search could find was parsed, this counter object was converted to a CSV file using Python's built-in CSV module. This process was done for each individual city and state so that the data could be stored separately.

B. Developer Languages Algorithm

To gather the developers first language as well as their most recently used language, a perl script was created to traverse the table responsible for commit data in the World of Code database. This table is split into one hundred separate binary files that need to be combed through to gather each developer's earliest and latest commits. For each file in the table an output file was created containing each developer's earliest and latest commit for that file. It is not guaranteed that all of each developer's commits will all be congregated in one of the files, so after the gathering process some cleaning needs to take place. To clean this data we congregated all of our data into one file and sorted it. This produced a single file with 677,801,240 entries of developer data. Since World of Code contains an estimated 50 million authors, we have quite a few duplicates. To clean this we created another perl script that scans each duplicate author for the earliest and latest commit and creates one entry for that author. After this round of cleaning we got a list containing 49,074,455 entries of developers and their earliest and latest commits. Now we need to map these commits to the language present in that commit. To do this we performed a mapping using World of Code from commit to files. We then classified the languages used by the extension of the files produced.

IV. RESULTS

Data from both job posting keyword searches and the world of code database was visualized and compared to determine how much, if any overlap there was between the languages individual programmers were using in their code repositories and the languages the industry values.
Bar and pie charts of the most highly sought-after languages were plotted and are presented.

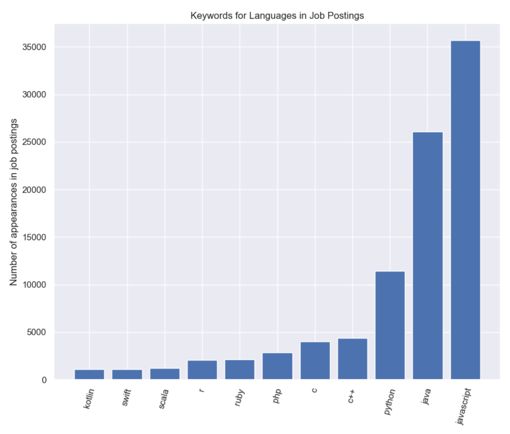

Figure 1. Top Ten Languages in Job Postings.

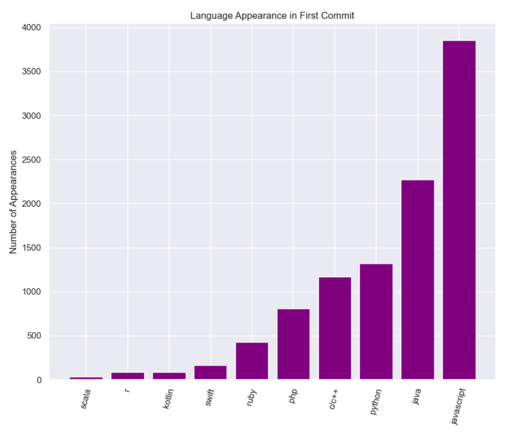

Figure 3. Top Ten Languages in First Commit

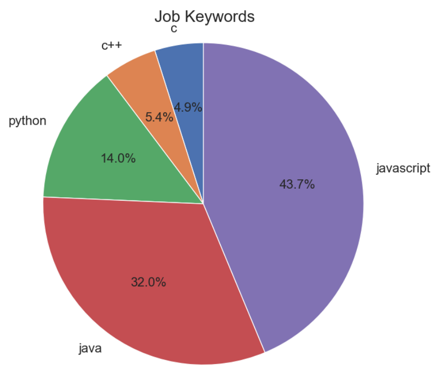

Figure 2. Top Five Languages in Job Postings.

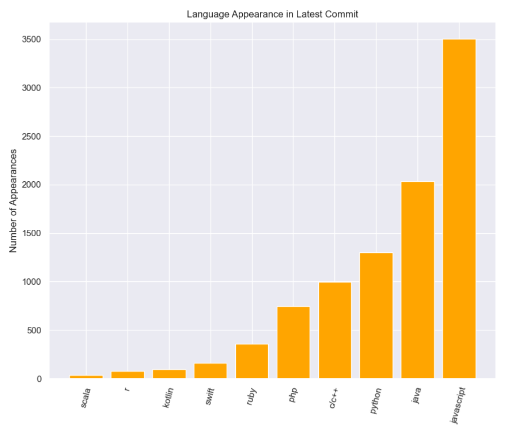

Figure 4. Top Ten Languages in Latest Commit

Clearly the industry favors JavaScript, closely followed by Java. With the increasing demand for mobile and web applications this stands to reason.

For the data pertaining to the languages present in commits, bar charts were generated to compare and contrast the first known commit of an author to the last known commit of the author. Those charts demonstrate that there is very little shift in the overall proportion of languages accounted for from earliest known commit to the latest.

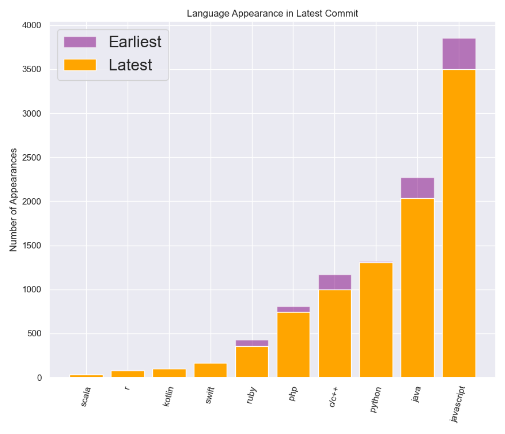

Figure 5. Earliest and Latest Commits Overlaid

Furthermore, these charts illustrate that not only do developers tend to not shift the languages that they use from earliest to latest commit, but that the languages that are present in commit histories are very closely correlated with the languages that the industry is searching for in job postings. In fact, the top five languages present in both data sets very closely match.

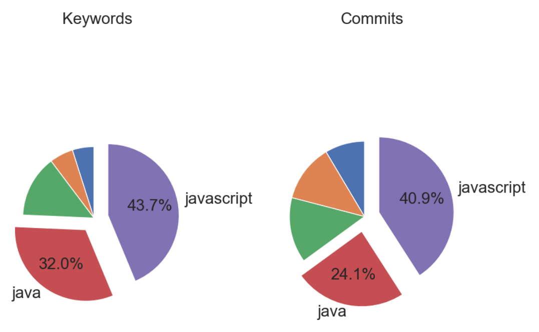

Figure 6. Prevalence of Java and Javascript Across Both Data Sets

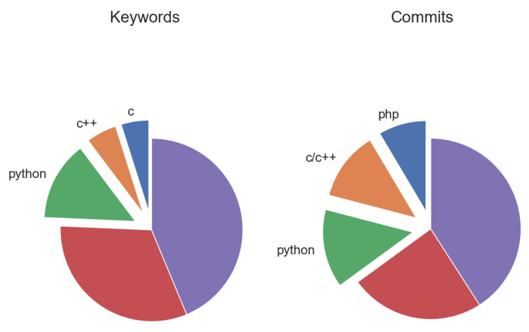

Figure 7. Other Languages Present in Both Data Sets

V. ISSUES ENCOUNTERED

When comparing the number of first commits versus the number of latest commits, the total number of commits is different. This can possibly be attributed to the methodology used in analyzing the data. Instead of comparing the languages used in first and last commits of a user, perhaps a better option would have been to instead compare the differences over time. This would also paint a clearer picture of the trends of which languages are used, and show the rate of change with more granularity.

At one point, in order to validate our assumption that the first commit of a programmer would use their first language, we contacted Github users through email. This was a misstep, and many mistakes were made because of this. For one, we had stored the list of email addresses we were contacting publicly. Having aggregated user data stored in one place publicly is bad form, as this information could be used as a resource for spam emailing en masse. We had also inadvertently violated Google's terms of service. We had created multiple email addresses using Gmail in order to send the emails at a faster rate. This violated Google's anti-spam policies.

VI. FUTURE WORK

Utilizing two commits to gauge a developer's progression when it comes to language use is not a very viable option. In the future it would behoove the group to make use of a timeline of commit histories for users to better assess the progression of their programming development. Likewise, it would be beneficial to perhaps utilize some sort of machine learning model to make some sort of assessment surrounding a developer's 'employability' by inferring this information from their commit history and other such statistics.

It would also be useful if the group were to follow the proper channels to contact the developers in an ethical way to verify the data acquired through World of Code. World of Code is an amazing asset, but data backed by a survey is always nice to have.

VII. TEAM RESPONSIBILITIES

A. John Carmack
   1) Data Cleaning
   2) Visualization
   3) Documentation

B. Dylan Lomax
   1) Data Acquisition (Developer)
   2) Data Cleaning
   3) Surveying

C. Lillian Coar
   1) Data Cleaning
   2) Documentation
   3) Floating Responsibilities

D. Brandon Mingledorff
   1) Data Acquisition (Industry)
   2) Data Cleaning
   3) Surveying

VIII. TIMELINE

- 2 weeks (October 8th)
    - Finalization of data sources and tools to be used for gathering and visualization, setting up access where necessary.
- 2 weeks (October 22nd)
    - Gathering of data using sources and tools chosen
- 1 week (October 29th)
    - Code testing and bug fixing, quality assurance of data to make sure it is ready for visualization
- 2 weeks (November 12th)
    - Visualization of data using tools chosen
- 1 week (November 19th)
    - Creation, practice of final presentation
- November 23rd
    - Final presentation
- 2 weeks (December 6th)
    - Creation, finalization of final paper