Chronological Series Sampling and Density-Based Estimation of the Total Number of Active Users on Github

Ashley M. Schuliger
Department of Computer Science, Worcester Polytechnic Institute, amschuliger@wpi.edu

Additional Keywords and Phrases: Github, Sampling, Estimation

1. INTRODUCTION

Github is a version control platform that is widely used by members of the software engineering community. They claim to currently host over 56 million software developers, with users constantly joining daily. Even though many users have been created on Github, there is a separate statistic that measures the number of users that are considered active. Users are considered to be inactive if they do not show any activity on Github for the duration of a month or more. The total number of active users is an important statistic for management purposes. Understanding the volume of users would help Github to develop ways to effectively store their users and manage their interactions with other users. However, Github does not share this information publicly. Calculating this information is not a simple task due to the many complexities of the user ID space. For starters, the user ID space itself is enormous, with the maximum active user ID being approximately 79,370,081 as of February 20, 2021. Attempting to exhaustively count the number of active users on the site would be extremely time costly and ineffective. In addition, even though Github provides a search API with functionality to determine the number of active users in a specified interval, Github has a ratelimit of 60 API requests per hour with only 30 responses returned per request. These limits prevent us from exhaustively calculating the total number of active users on Github efficiently. Current work on sampling and estimating the lengths of similar ID spaces utilizes the sampling of specific intervals by exploiting special properties of a search API. For example, a study on estimating the number of videos uploaded on YouTube utilized a prefix-based search technique that allowed them to randomly sample IDs with prefixes of specific lengths using the YouTube search API [1]. While this technique worked effectively for the YouTube video ID space, it does not work for our Github user ID space. YouTube randomly generates IDs for each of its videos, whereas Github assigns user IDs in chronological order. This ID assignment method may result in varying densities of active users throughout the entire user ID space. Thus, a random sampling and estimation technique such as the one used for estimating YouTube IDs will not work in our case since it does not take into account varying densities across the space. This density property of the Github user space lays down the foundation for a sampling and estimation technique best suited for the estimation of active user IDs. In the subsequent chapters, we propose both a sampling and estimation method that uses density-based measurements to calculate the number of active user IDs on Github. Our estimation technique is similar to that used in Dynamic Range Calibration in that it calculates the total number of user IDs based on the density of individual intervals throughout the entire space [2]. We provide the methodology for this technique and prove that our proposed estimator is unbiased. We then execute our sampling and estimation techniques on two subsets of the Github user ID space and perform a validation of our methods. Once validated, we implement our methods on the real Github user ID space to calculate an estimation of approximately 76,507,369 for the true number of active user IDs on Github.

2. METHODOLOGY

In this section, we provide a methodology by which to sample and estimate the number of active users on GitHub. We also provide a proof to show that our estimator is unbiased.

A. Chronological Series Sampling

One function of Github's API that we can utilize for sampling is the Search by ID API. With this functionality, we can provide Github's API with a random ID (any integer, z_i where $0 \leq z_i$), and Github will return a list of up to 30 active user IDs, starting with the given ID. This technique can result in uneven interval lengths, as some intervals may contain more active users than others. For example, if we provide the Search API with user ID $z_i = 50$, we may retrieve 30 responses from user IDs 50 to 100, indicating that there were 20 inactive user IDs in the interval. In contrast, another interval starting with user ID $z_i = 70{,}000$ may retrieve 30 responses from user IDs 70,000 to 70,030, indicating that there were 0 inactive user IDs in the interval. It is important to understand

this distribution of the user IDs to accurately sample the user ID space. Another useful function of Github's API is "Search by date created". This functionality allows us to indicate a created date where we would like our query to start searching for user IDs. The API will return a list of up to 30 active user IDs that were created on or after the given date. This API tool is important for estimating the range of the user ID space. In using this functionality, we were able to calculate a maximum user ID of 79,370,081 as of February 20th. Since user ID $z_i = 1$ is still active, our range of the entire space, denoted as r(X), is equal to 79,370,081. This information was crucial to our sampling method as it allowed us to better define our intervals given our time and space limitations. Based on these properties of Github's Search API, we initially proposed using a random-interval sampling technique similar to that used in Random Prefix Sampling to estimate the total number of YouTube videos [1]. However, an interesting obstacle arises when considering the way that users are assigned IDs on Github. While YouTube assigns video IDs randomly, Github assigns IDs in chronological order. In this way, the initial user of the platform was given a user ID of 1, whereas the most recent user of the platform was given a user ID well above 79,000,000. Not only is this assignment not random, but it also creates bias in terms of how dense certain intervals of the data space are. There is a higher likelihood of users being active in later user ID intervals since the users were created more recently than those at the beginning of the space. In terms of sampling, this becomes problematic as we aim to create a sampling method that is fair and summarizes the entire ID space. Along with this, sampling intervals that overlap with one another can cause issues and prevent our estimator from being fair and accurate. These obstacles justify the design of the sampling method used in this case. In our sampling technique, we start by querying user ID $z_i = 0$ to obtain the first interval of our sample. From here, we store the calculated density of the interval and determine the next ID to query by taking the last accessed active ID (i.e. the last user ID in the given interval) and adding a constant, c, to it. In this way, we can ensure that no interval is overlapping, and we avoid disproportionate region sampling of the user ID space by sampling methodically interval by interval. On another note, the starting ID can be adjusted in order to increase the randomness of sampling, but the general technique is needed to ensure fairness.

B. Density-Based Estimation of Active Users on Github

After using the sampling method discussed previously, we chose an unbiased estimator for $\widehat{N}$ that closely follows the estimator proposed in the estimation of GeoSocial Data [2]. In this problem, the goal was to estimate the total number of venues in a large-scale region. Similar to the Github user space, geographic regions are made up of subregions with varying densities. Due to this, building an estimator must account for the density of a sample to provide an unbiased estimation of the number of venues in a region. In our case, we attempt to remove bias by finding the average density of the entire user ID space and calculating the total number of active Github users from there. Density is defined as

$$d(x_i) = \frac{f(x_i)}{r(x_i)},$$

where $f(x_i)$ is the total number of responses in an interval, and $r(x_i)$ is the length of the sampled interval. We then can calculate the average density for the space by taking the sum of all interval densities and dividing it by the total number of sampled intervals. From here, we notice that the total number of active Github users can be modeled as

$$N = d(X)r(X),$$

where d(X) represents the density of the entire space, and r(X) represents the length of the entire space. Next, we can model an estimator for the total number of active Github users, as stated in Theorem 1.

THEOREM 1 (Estimator of the Total Number of Active Users). Given m samples x_i ($1 \le x_i \le m$) by evenly querying non-overlapping ID intervals with varying lengths, denoted by $r(x_i)$, and number of response, denoted by $f(x_i)$, we have the unbiased estimator

$$\widehat{N} = \frac{r(X)}{m} \sum_{t=1}^{m} \frac{f(x_i)}{r(x_i)}$$

where r(X) the range of the user ID space, r(X).

PROOF. We start by considering how user IDs are generated by Github. As mentioned previously, Github creates user IDs in chronological order, such that the most recent user has the maximum ID and the oldest user has user ID $z_i = 1$. Due to this, it is fair to suggest that earlier intervals of user IDs will have a lower density than more recent intervals. As such, we must calculate our estimator in terms of

$$d(x_i) = \frac{f(x_i)}{r(x_i)}$$

where $d(x_i)$ represents the density of a sampled interval. From here, we can find the expectation value of a single density measurement. Since our sampling method forced us to collect samples from the entire range of user IDs, we know that our collected densities can be used to calculate the average density of the space. Thus, we can conclude that the expected value of a given density interval is the average density across the entire space, such that

$$E[d(x_i)] = \frac{N}{r(X)}$$

where N is the ground truth for the total number of active Github user IDs. From here, the expectation of $\widehat{N}$ satisfies the following:

$$E[\widehat{N}] = E[\frac{r(X)}{m} \sum_{t=1}^{m} \frac{f(x_i)}{r(x_i)}]$$

Since r(X) and m are constants in this equation, we can write

$$E[\widehat{N}] = \frac{r(X)}{m} \sum_{t=1}^{m} E[\frac{f(x_i)}{r(x_i)}],$$

which can then be simplified to

$$E[\widehat{N}] = \frac{r(X)}{m} \sum_{t=1}^{m} \frac{N}{r(X)}] = \frac{r(X)Nm}{mr(x)} = N$$

as previously mentioned. Thus, this proves that $\widehat{N}$ is unbiased.

3. EVALUATION & RESULTS

This section will cover the performance of the above-mentioned sampling and estimation algorithms on real data. We will first discuss the results of these algorithms using two validation sets to provide a baseline for further estimations. We exhaustively count the number of active users in these set intervals and use our estimator to estimate the calculated number of active users in the interval. Next, we perform an estimation on the entire real user ID space of Github and provide an estimation for the total number of active Github user IDs.

A. Validation of Estimation Technique

We start by providing validation of our technique using two different intervals from the user ID space to provide a baseline for our estimation. We exhaustively searched our chosen intervals to find the total number of active user IDs in the given space. The first chosen interval starts at user ID $z_i = 0$ and ends at $z_i = 38,203$, while the second randomly chosen interval starts at user ID $z_i = 53,838,137$ and ends at $z_i = 53,867,840$. These subsets were purposefully selected from different regions of the user ID space to provide an estimation using areas of differing densities. Since our user space contains varying densities, our validations will be more reliable if we use subsets of differing densities than if we use subsets of similar densities. Figure 1 and Figure 2 below show off the results of implementing our sampling and estimation methods on our subsets. In this implementation, we randomly chose an order to sample our API requests and calculated a new estimation after each request was made using the cumulative density at that instance. This allowed us to collect estimations at varying numbers of sampled intervals. We can sample this way since the intervals are small enough to assume a consistent density across the entire interval space. We iterate through this process of randomly choosing the order and calculating an estimation approximately 40 times to create the plots below.

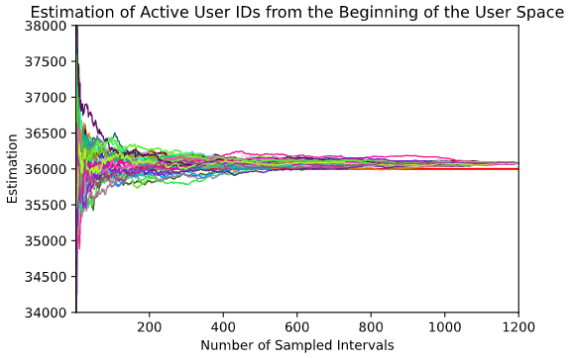

Figure 1: Validation of our sampling and estimation methods on the interval from the beginning of the user ID space

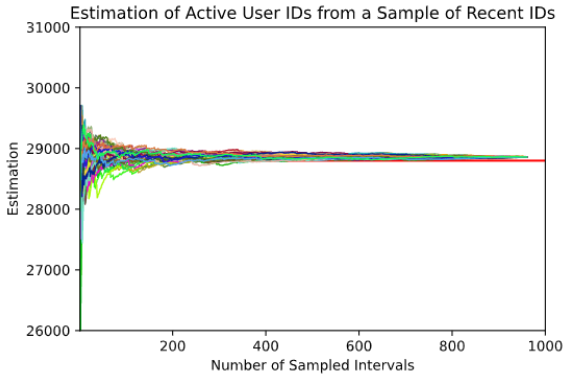

Figure 2: Validation of our sampling and estimation methods on the interval from later in the user ID space

As shown in the figures above, both of our validations converged early to the ground truth. Thus, we can conclude that our estimator accurately calculated the approximate number of active users in the given user space. These results provide a baseline for how our estimator and sampling method will perform on the real Github user space.

B. Estimating the Total Number of Active Github User IDs

Now, we estimate the total number of active Github user IDs using the techniques described in previous sections. Our sampling technique involved gathering data from intervals starting at user ID $z_i = 1$. We chose our sampled intervals by adding c = 90,000 to the last accessed value in the current API request. We ran approximately 60 API requests each hour through iteration and chose to store the cumulative sum of densities, the last accessed value, and the number of sampled intervals (i.e. number of API requests) after each iteration of 60 requests. We continued to run these iterations until we hit the maximum active user ID. As mentioned in previous sections, we estimated the maximum ID of the user space to be 79,370,081; however, after performing this sampling technique, we found that our true maximum user ID is 79,618,051. We use this calculated maximum value to estimate the true number of active users in the Github user space. Figure 3 below shows the relationship between the number of intervals versus the estimation with one iteration of our sampling method.

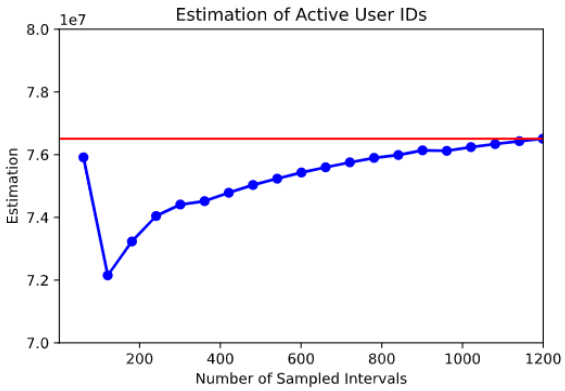

Figure 3: Implementation of our sampling and estimation methods with the estimation at each interval compared against the number of sampled intervals

Notice that, with the exception of the first sample, the estimation of active users increases and converges to our ground truth as more intervals are sampled. This aligns with our suspicion that chronologically-earlier samples have lower densities than recent samples. As a result of this implementation, we estimate the total number of active users on Github to be approximately 76,507,369. We believe this estimation to be close to the ground truth since our validations discussed in the previous section confirmed the strong performance of this estimation model.

4. DISCUSSION

Based on our validation and estimation results, our techniques provide a strong estimation for the number of active user IDs in the Github user space. However, there are ways we can improve upon our sampling and estimation methods. In hindsight, we should have stored the densities at each request so that we could randomly sample the sampled intervals, rather than simply storing the sum of densities after 60 requests. In the future, we plan to use the same sampling method as described above while also storing the calculated density of each API request to obtain a dataset to further sample. From here, we will perform multiple iterations of random sampling on this subset of the user ID space and calculate the estimations with these samples. We also plan to calculate the estimations with different sample sizes (i.e. number of requests per sample). By doing this, we can average the calculated estimations to obtain an accurate estimation of the total number of active users in the Github user ID space. Another way we can improve is in our calculation of the length of an interval. Currently, we measure the length of the interval as the last ID in the response subtracted by the first ID in the response. However, there is a possibility that the ID we use in the Search API is an inactive ID. In this case, the Search API will find the next active user and start the query from there. Thus, in the future, we should calculate the length of the interval as the last ID in the response subtracted by the ID used for the query. This will give us a more accurate density for the space and more information about the space as a whole.

5. CONCLUSION

In this paper, we proposed chronological series sampling using the Github Search API and density-based estimation as a way to create an unbiased estimation of the total number of active users in the Github user ID space. Implementation on a validation set as well as on the real Github space show the unbiased nature and accuracy of our estimator and sampling method. We also provide a discussion on ways to improve the overall accuracy of our methods through further sampling of our sampled intervals of the Github space. The proposed sampling and estimation methods provide a strong baseline

from which to expand upon to more accurately estimate the total number of active user IDs on Github. In future work, we plan to explore ways to improve these sampling and estimation methods to better utilize the information provided by our samples and provide a more accurate estimation of the total number of active user IDs in the Github user space.

REFERENCES

1. Zhou, J., Li, Y., Adhikari, V. K., & Zhang, Z. L. (2011, November). Counting youtube videos via random prefix sampling. In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (pp. 371-380).
2. Li, Y., Steiner, M., Bao, J., Wang, L., & Zhu, T. (2014, March). Region sampling and estimation of geosocial data with dynamic range calibration. In 2014 IEEE 30th International Conference on Data Engineering (pp. 1096-1107). IEEE.