An Emperical Analysis on User Interests on GitHub

Jiacheng Pan

November 29, 2014

1 Introduction

GitHub is a very popular platform for developing software in collaboration nowadays. At the same time, with many features provided by the website, such as following a user, starring and forking a repository, people can use GitHub as an information platform to share and obtain new ideas all over the globe.

In this paper, investigations are made to explore 1) how users' interest towards repositories looks like; 2) how repositories are related to each other based on the context of users' interest; 3) how users' interest as well as the relationship between repositories change over time.

2 Related Work

Some recent works have examined the conditional probablities as well as the correlations between programming languages used by users and the language pairs [4, 10].

Also based on GitHub, the relatioship between repository collaboration networks has been examined and visualized, in accordance with the programming languages being used [5, 8].

User communities based on primary programming languages and "follower" relationships were explored in [3, 14]. A visualization of the bipartite graph of organizations (groups of users) and languages [9] won third place in the 2012 GitHub Data Challenge.

Important nodes in the one-mode project-project and user-user networks, weighted simply by number of common links, were identified in [11].

Also recent models of network evolution capturing the growth process incorperating 1) constant average degree assumption; 2) slowly growing diameter assumption are discussed in [2, 12, 13].

3 Data Set

3.1 Data Collection

GitHut published their timeline data since 2012 at www.githubarchive.org [6], and the same datasets are as well queryable via Google BigQuery [7], through which all the raw data used in this paper are collected. In particular, the space of public watch events, i.e. starring a repository when people finds it interesting, on GitHub between January 2012 to October 2014 are collected and aggregated every quarter. At the same time, the main language used and the accumulated star number of each repository of interest are collected. In order to reduce computational complexity, only the watch events towards the repositories that have more than 1000 accumulated stars by the end of each quarter are collected.

3.2 Repository Interest Graph

In order to explore how users' interest may look like, this paper tries to observe a graph with repositories being the nodes, while the count of starring users of both repositories being the edges.

Through this graph, which is derived from a bipartite graph between users and repositories, it may be easy to find out what else repositories may be liked by users when a certain repository is starred. And therefore to further estimate the relationship between repositories based on such an interest context.

In detail, we define graph nodes as the repositories being starred, when the repository has more than 1000 accumulated stars by the end of each quarter. Also, we connect two nodes, i.e. repositories, when a user has starred both of the repositories within the same quarter. And the total number of such users is marked as the edge weight.

Since we intend to explore how the repository interest graph may change over time, eleven such graphs are built thoughout the timeline data, snapshotting users' behaviours within each quarter.

4 Analysis

4.1 Static Observation

In this section, a snapshot network is analyzed, by viewing the layout and colouring of the graph, as well as by using network metrics.

4.1.1 Visual Analysis

We first select the Y13Q2 (Year 2013, Quarter 2nd, April 2013 – June 2013) dataset to analysis a network snapshot of the repository interest graph.

Using Gephi, with the Force Atlas layout algorithm, the graph is shown in figure 1(a), coloured with Moduality Class cacluated by Gephi, and sized in accordance to the accumulated star number of the repositories by the end of the quarter. At the same time, the same graph is shown in figure 1(b) using language colouring, i.e. colouring the repositories regarding the main language they use.

Comparing these two plots, it is easy to find out that the community distribution correlates with the language. And from this observation, it is also understandable to deduct that people are likely to star repositories of languages they are famaliar with.

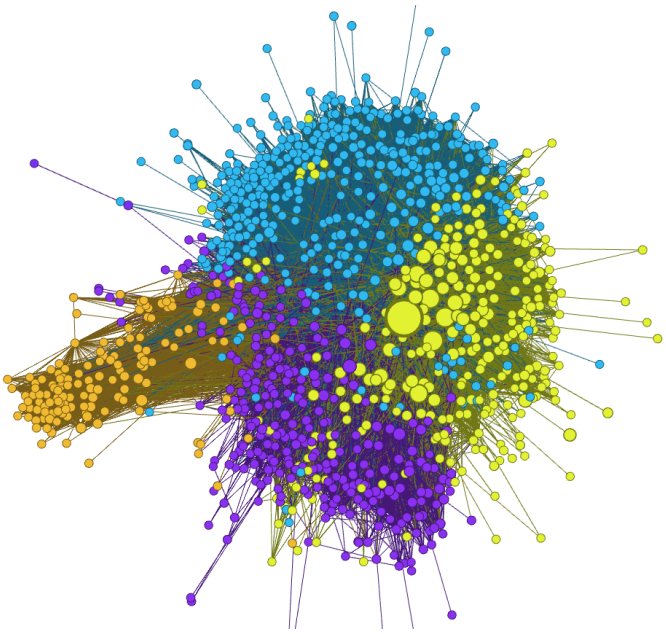
Also from the both plots, the Objective-C cluster stands out of the main cluster, forming a relatively lonely community (figure 2). This shows 1) users who likes objective-c repositories are likely to be interested in objective-c repositories themselves; 2) users who likes objective-c repositories are less likely to be interested in Ruby or VimL (the purple nodes near the Ruby cluster) or PHP, and so on. And therefore we may further guess that objective-c users may be more distant and isolated from the world of web programming, which involves languages like Ruby, PHP, and Javascript etc., in terms of their interests.

4.1.2 Analysis of Network Metrics

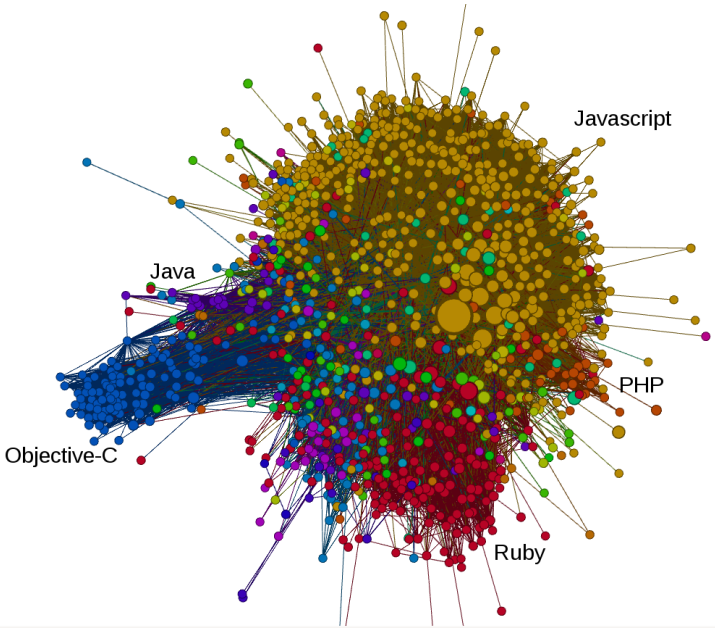
To deeper understand the Y13Q2 dataset, degrees (weighted and unweighted) as well as betweenness (weighted and unweighted) are caculated and the top 10 repositories of each metrics are shown in tables 1, 2, 3, 4.

In terms of the degrees, it can be seen that Javascript repositories are likely to be "common interests" of many users, because degrees, weighted or not, generally shows how commonly the repository is starred by users along with other repositories. And weighted degrees may better show the popularity amongst the users, while degree more focuses on the inter-repository relationship.

In fact, the observed phenomenon is reasonable when considering the real situation: 1) Javascript is an easy-to-get-started language with relatively flat learning curve, thus liked or touched by many users from different fields; 2) The number of Javascript repositories that have over 1000 stars is already high, and people are likely to be interested in the language they know.



(a) Dataset Y13Q2, coloured with Moduality Class



(b) Dataset Y13Q2, coloured with Languages Used by Repositories

Figure 1: Dataset Y13Q2, Colouring with community and language

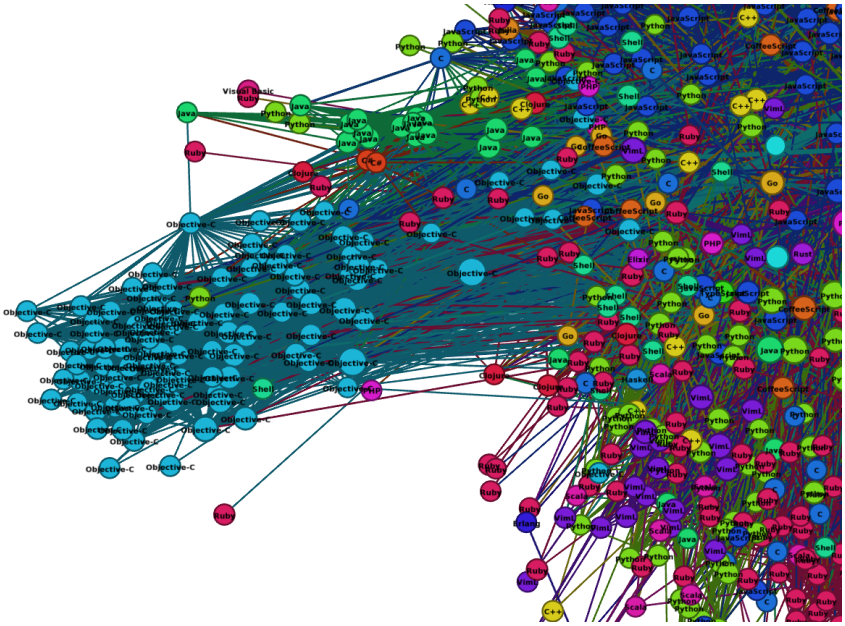


Figure 2: Dataset Y13Q2, Zoomed into Objective-C cluster

However, when comparing top-degree repositories with top-star repositories, it turns out that repositories with high star number may not necessarily be the top-degree repository. For instance, twitter/bootstrap has the highest star number (52333), whereas yui/pure only has 4231 stars, but it is shown twitter/bootstrap is not as common-liked as yui/pure, and several other repositories. This phenomenon may be due to temporal factors. For example, twitter/bootstrap is a long-published repository while yui/pure is much younger than twitter/boostrap. Also the name of twitter may attract more people since it is a very popular platform, while yui, an open-source Javascript and CSS library, is less well-known to common people.

While in terms of the betweenness, in particular the weighted betweenness, the top repositories tend to be useful utilities, therefore more popular amongst people who use different languages. For instance, the unconed/TermKit is a terminal / command application, github/gitignore a collection of .gitignore templates, and progrium/localtunnel a local-sever-to-internet exposer, all share the same feature that they can be useful to any programmer or GitHub user. And actually, this phenomenon well collaborate with the meaning of betweenness centrality.

Repositories	Degrees
yui/pure	1252
heelhook/chardin.js	1239
topcoat/topcoat	1235
angular/angular.js	1233
dypsilon/frontend-dev-bookmarks	1232
hakimel/Ladda	1231
jakiestfu/Snap.js	1230
FortAwesome/Font-Awesome	1229
zmoazeni/csscss	1228
twitter/bootstrap	1227

Table 1: Unweighted Degrees

Repositories	Degrees
yui/pure	49033
heelhook/chardin.js	39376
hakimel/Ladda	38779
topcoat/topcoat	35796
dypsilon/frontend-dev-bookmarks	35491
jakiestfu/Snap.js	34491
helios-framework/helios	30749
angular/angular.js	30733
twitter/bootstrap	30589
dimsemenov/Magnific-Popup	30114

Table 2: Weighted Degrees

Repositories	Betweenness
yui/pure	3020
angular/angular.js	2605
gitlabhq/gitlabhq	2553
h5bp/html5-boilerplate	2494
github/gitignore	2331
mneorr/Alcatraz	2327
heelhook/chardin.js	2305
toursprung/TSMessages	2149
usablica/intro.js	2146
designmodo/Flat-UI	2077

Table 3: Unweighted Betweenness

Repositories	Betweenness
unconed/TermKit	2093
gotosleep/JASidePanels	1945
progrium/localtunnel	1755
usablica/intro.js	1621
github/gitignore	1618
sobri909/MGBox2	1562
square/crossfilter	1559
tonymillion/Reachability	1552
Simbul/baker	1536
mneorr/Alcatraz	1531

Table 4: Weighted Betweenness

4.2 Dynamic Analysis

4.2.1 Visual Observation

Figures 3 and 4 are present to show the general trend of GitHub repository interest graph evolution over time. It can be seen that generally, language and community are closely correlated over all the time, not only for Y13Q2 data as analyzed previously.

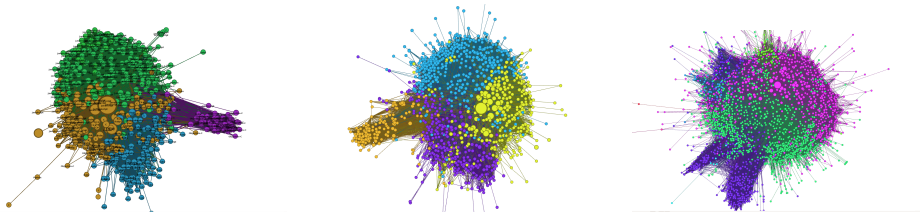
Also, as time goes by, the number of language clustering is as well increasing – from Y12Q2 with 3 big clusters, to Y14Q2 with 6 clusters. And this process is actually a gradual evolution, for example, Java cluster was almost un-obsevable in Y12Q2, and a small cluster was established near Objective-C in Y13Q2, and in Y14Q2, the Java cluster already forms another out-standing cluster just as Objective-C cluster – they form two sharp strikes in the graph plot, as shown in figure 5.

This increase of language cluster possibly shows that GitHub is actually getting more and more popular among programmers using different languages. And they are increasingly likely to share their codes and get aquainted with other people's work on GitHub.

Diving into the Y14Q2 language, we can still find out that the largest

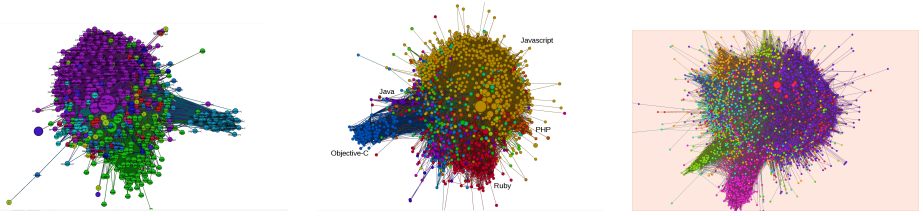
community or language cluster is still Javascript. However, as is shown in a zoomed version in figure 6, there are many red dots inside the Javascript cluster, which are CSS repositories.

After taking a deeper investigation, two factors may be contributing to this phenomenon: 1) these CSS repositories are indeed using CSS as its highest portion of languages used, while Javascript usually just lies next to CSS, therefore the repositories still is highly relying on Javascript; 2) GitHub may have enhanced their language detection routines to seperate CSS from Javascript after Y13Q2, because from the data, it is strange that neither Y13Q2 nor Y12Q2 has CSS repositories included.



(a) Y12Q2 (b) Y13Q2 (c) Y14Q2

Figure 3: Community Colouring Graphs



(a) Y12Q2 (b) Y13Q2 (c) Y14Q2

Figure 4: Language Colouring Graphs

4.2.2 Analysis of Network Metrics

To quantatively analyze the network change, we caculated average path length and density of 11 graph snapshots, as well as the node and edge count trend, shown in figure 7.

From the plots of average path length and density, it is hard to conclude any increasing or decreasing trend. Rather, the general trend looks to be scattering around a certain value.

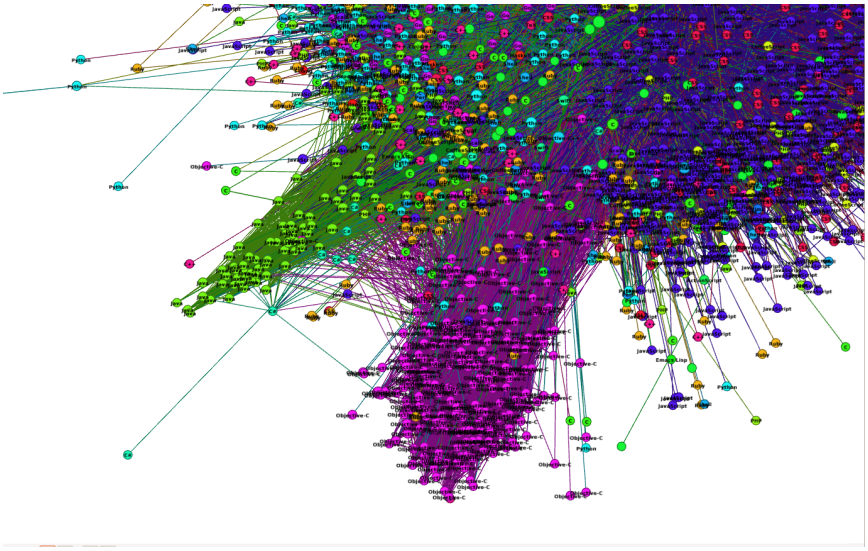


Figure 5: Y14Q2 Zoomed Version of Language Colouring Graph – Java and Objective-C

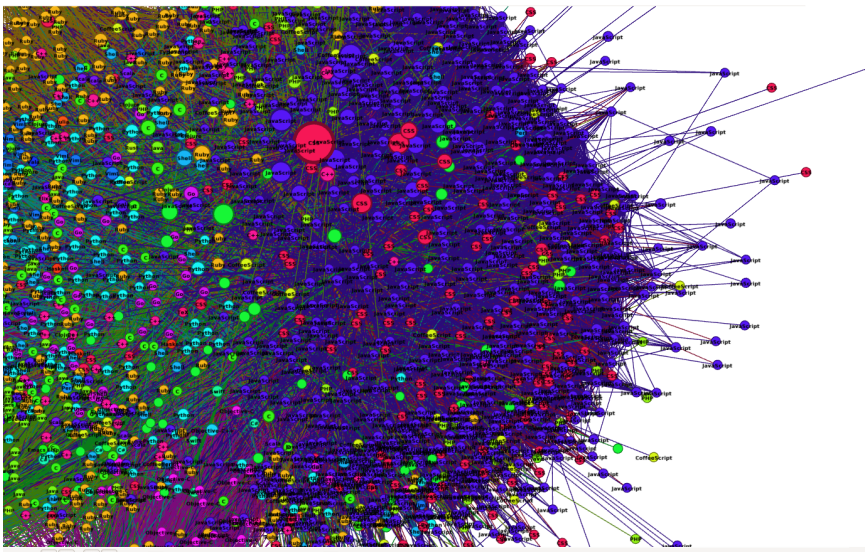


Figure 6: Y14Q2 Zoomed Version of Language Colouring Graph – Javascript

However, when looking into the node and edge count trends, it is obvious to tell a linear or polynomial increase.

Therefore, it shows that for the GitHub repository interest graph, although it increases in size quickly throughout time, the density or average path length is not remarkably increased. That is, repositories are almostly as tightly connected to each other as before, based on the context of users' interest, and therefore users of GitHub are possibly diverse in interest, as well as the repositories.

Also, when focusing on certain repositories' weighted degree, it can also be found that some repositories were popular once but then tend to be unknown, while others may be continuously popular among people, though the starring people of each quarter may not be that many.

For instance, in Y14Q1, there is a repository naming sahat/hackathon-starter. It achieved a 83571 weighted degree at that time, ranking the first amongst 2180 repositories of interest. However, in Y14Q3, it only obtained 5447 weighted degree. Although still relatively high, its great days are no more – it's ranking 430 out of 2998 repositories.

In contrast, repository twbs/bootstrap achieved 35419 weighted degree in Y14Q1, ranking 44, and still has 32198 weighted degrees in Y14Q3, ranking 67.

Therefore, different repositories have different popularity trend from this observation.

5 Conclusion

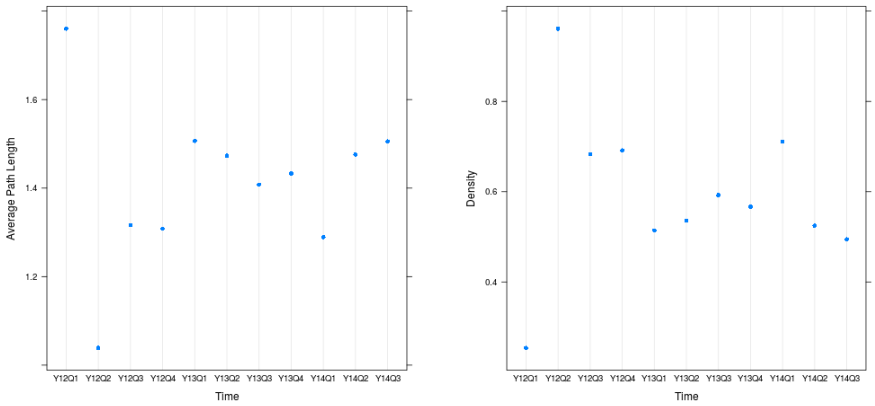
In this paper, analysis is made to explore, on GitHub, how users' interest towards repositories are like, how repositories are related based on such context, and how such relationship may evolve over time.

The conclusion, based on both visual observations and network metrics analysis, is:

1. users tend to be interested in the repositories of the languages they are familiar with;

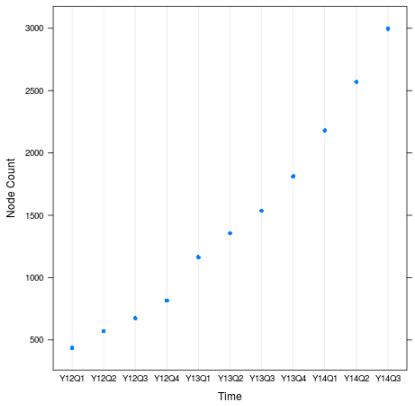
2. repositories of same languages or languages of the same function are likely to be clustered together;

3. over time, more and more users with different language preferences are likely to migrate into GitHub, thus creating more and more language communities, exposing clearer language community characteristics;

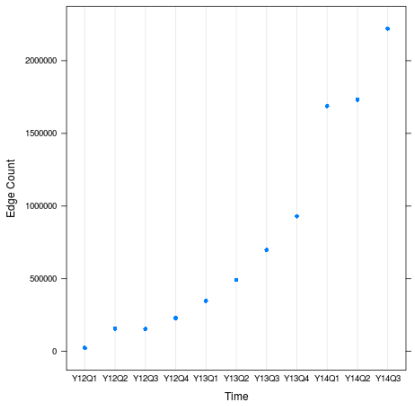


(a) Average Path Length

(b) Density



(c) Node Count



(d) Edge Count

Figure 7: Network Metrics Trend

4. over time, the repository interest graph is increasing in size, but its density, i.e. inter-repository relationship, tends to keep tight;

5. over time, some repositories got really high popularity but later falls into less known, while some continuously have mild starring number.

However, since the data sampled are only limited to repositories with high star number, the conclustions made in this work may not be accurate, for even more repositories, which may be popular among smaller groups, are not included into the examination. Further work need to be done if a deeper investigation is to be made. Also, starring is only one aspect showing users' interests, whereas forking, watching, user following are also essential to be applied for analysis. The data used in this work is therefore limited to certain extent. More comparisons may be needed to better analyze the GitHub network.

6 Acknowledgement

This paper is developed as a course project on social network analysis[1]. All source codes as well as plots are available at https://github.com/jiachengpan/sna.

References

[1] Lada Adamic. Social network analysis. https://www.coursera.org/course/sna, 2013.

[2] R. Albert, H. Jeong, and A.L. Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

[3] Franck Cuny. Github explorer. http://lumberjaph.net/graph/2010/03/25/github-explorer.html, March 2010.

[4] Brian Doll. Programming language correlation dataset. https://gist.github.com/briandoll/e0637fff9c8eec988528, April 2012.

[5] Corey Ford. Programming languages and collaboration communities on github. April 2013. Course project report https://github.com/coyotebush/github-network-analysis.

[6] GitHub Archive. http://www.githubarchive.org/.

[7] Google BigQuery. https://bigquery.cloud.google.com.

[8] Joseph Marrama and Tiffany Low. Social coding: Evaluating Github's network using weighted community detection. Course project report.

[9] Eduarda Mendes Rodrigues. GitHub data. http://labs.sapo.pt/networds/2012/05/09/github-data/, May 2012.

[10] Akshay Shah. Language use on GitHub. http://datahackermd.com/2013/language-use-on-github/, February 2013.

[11] Ferdian Thung, David Lo, and Lingxiao Jiang. Network structure of social coding in GitHub. 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013.

[12] Andrei Broder und Ravi Kumar. Graph structure in the web. 2000.

[13] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

[14] Nicholas M. Weber. Combined methods, thick descriptions: Languages of collaboration on Github. *Proceedings of the American Society for Information Science and Technology*, 49(1):1–4, 2012.