Table of Contents

[Problem Overview and solution 3](#_Toc361324483)

[Problem Statement 3](#_Toc361324484)

[Dataset 3](#_Toc361324485)

[Solution 3](#_Toc361324486)

[Algorithm 3](#_Toc361324487)

[Implementation 3](#_Toc361324488)

[Genuineness of a user’s impact 3](#_Toc361324489)

[User Dynamics 4](#_Toc361324490)

[Code setup 5](#_Toc361324491)

[Generated Growth Curve Plots 5](#_Toc361324492)

[Plot properties 7](#_Toc361324495)

[Observations 7](#_Toc361324496)

[Dataset statistics 8](#_Toc361324497)

[Appendix 9](#_Toc361324498)

[WatchEvent 9](#_Toc361324499)

[FollowEvent 11](#_Toc361324500)

# Problem Overview and solution

## Problem Statement

Find the relationships between the watchers/committers with the popularity of a repo. How does the push/watch event of a highly popular user affect the growth curve of a GitHub repository?

## Dataset

The dataset is in the form of JSON dumps of GitHub activity of various repositories and users. Please find the sample files [**here**](https://github.com/geekypunk/GitHubTrends/tree/master/sampleJSONs)

## Solution

The solution attempts to find the correlations between user events on a repository and its effects on the growth curve. The approach is fairly straightforward.

### Algorithm

* First select the topmost users aka high profile users (defined having highest number of followers). This is information is obtained from [FollowEvents SQL dump](https://github.com/geekypunk/GitHubTrends/blob/master/sqlDumps/FollowEvents.sql), created using all the [FollowEvents](#_FollowEvent) in the JSON dump
* Then get all the [WatchEvents](#_WatchEvent) of each of these high profile users
  + For every WatchEvent by a user on a repository, get all the events on that repo till 24 hours after the WatchEvent.
  + Draw a growth curve for this repository based on the above 1 day events. The x-axis contains the timestamps represented as float values and y-axis has the number of watchers of the repo at the respective timestamps
  + We first determine the plot’s eligibility to be drawn. We draw a plot if the growth curve shows substantial deviation from its predicted curve. The predicted curve is drawn since the timestamp of the WatchEvent. The data prior to this event is used for prediction. Least squares polynomial fitting is used to draw the curve.
  + Then we determine the **genuineness of a user’s impact (**[explained later](#_Genuineness_of_a)**)** on that repo. If a user’s WatchEvent has actually caused the change in the growth curve, the plot of that repo is generated and stored in an image file
* Using the above generated growth curve, we also try to observe as how group dynamics work in social coding. Do some users act as flock when one of them starts watching a repo? See [**User dynamics**](#_User_Dynamics) section for more info

### Implementation

The python implementation for the above algorithm can be found [here](https://github.com/geekypunk/GitHubTrends/blob/master/python/highProfileWatchEvents.py).

### 

### Genuineness of a user’s impact

Since the change in the growth curve of a repository need not be influenced by a WatchEvent of a particular user, we try to estimate as how **genuine** his WatchEvent is. The algorithm used for this is as follows

* Get all the changes in watch counts of all the repos he has started watching. The initial and final watch counts differ by a timestamp of 1 day.
* Calculate the standard deviation of all these differences.
* If the standard is low “enough”, we deem the impact to be genuine, i.e the growth change has happened due to the user watching the repo

### 

### User Dynamics

This section explains the methodology used to observe user behavior in the social coding context. Do user operate in flock, when one of them starts following a repo? Which users most “connected”, i.e. the users most probable to start watching a repo, when one of them does?

#### Algorithm

* Using the growth curve generated from the above algorithm, we can create lists of chronological user sequences. Each sequence contains a list of usernames, which represents the all the users,in-order,who have started watching the repo after the first user in the list has.
  + For example for a repo, if the first WatchEvent was by a high profile user called **A**, and the subsequent watch events on the repo till 1 day after that were by **B C D and E**. The list would look like [A,B,C,D,E]
* So essentially we will have a list of such sequences.
* Now, to determine the user groups which are most “close”, we simply need to determine the subsequences which are most common among all the sequences in the list. We say sub-sequence and not sequence as the users need not appear in the exact same order in every sequence encountered, but just should be chronologically same. For example if A appears before C in a sequence, and same is in another sequence, it is safe to assume A’s influence on C. This assumption is only valid upto a certain distance between the two users. As the longer is the gap between two users in a sequence, lower is their “connectivity”, i.e lower is A’s influence on C.(**Currently working on accommodating this logic**)
  + This high co-incidence user groups is obtained by maintaining a map of all possible permutations, of all the sequences encountered so far as we iterate the list of sequences. Whenever we see an already seen permutation, we increment its occurrence count.

##### Sub-Algorithm

Input : List on chronological user sequences **S**

Output : Count map, **M, representing the occurrence count of all sub-sequences**

For all sequences Q in S:

For every possible permutation P of sequence Q in map M:

Check if P already present in M:

If True:

M[P]++ //Increase the occurrence count

#### Implementation

The python implementation for the above algorithm can be found [here](https://github.com/geekypunk/GitHubTrends/blob/master/python/getFlockUsers.py). The script takes two command-line arguments, first one is the size of user sets and second is how many such top sets.

For example,

If we need the top 5 users sets each of size 3 who show high connectivity, issue the command

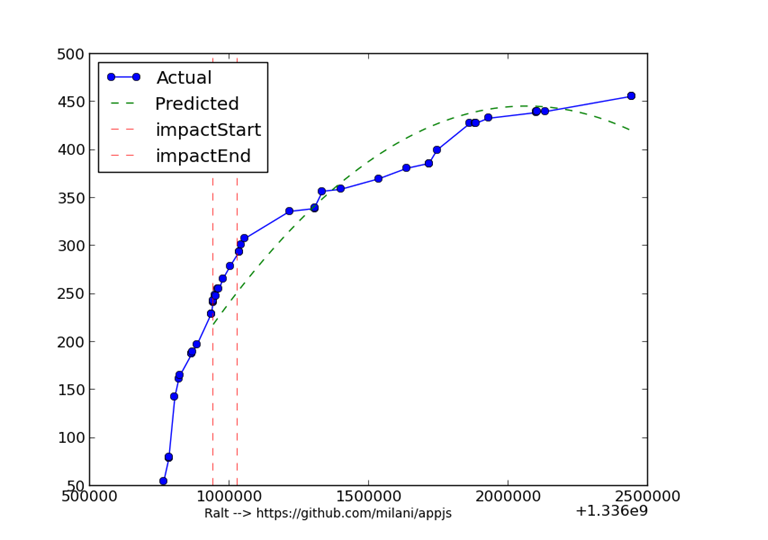
**python getFlockUsers.py 3 5**

# Code setup

* Put appropriate database credentials in def getDBConnection() function of [highProfileWatchEvents.py](https://github.com/geekypunk/GitHubTrends/blob/master/python/highProfileWatchEvents.py)
* Import the sql files from [here](https://github.com/geekypunk/GitHubTrends/tree/master/sqlDumps)
* Run [highProfileWatchEvents.py](https://github.com/geekypunk/GitHubTrends/blob/master/python/highProfileWatchEvents.py) **to generate plots**

# Generated Growth Curve Plots

Here is a sample of generated plot images of the repo.



## 

## 

## Plot properties

* X-axis indicates the timestamps
* Y-axis indicates watcher counts at respective timestamps
* The first red line(impactStart) indicates the timestamp of the high profile WatchEvent
* The second red line(impactEnd) is 24 hours after impactStart
* The text below the plot is of the format
  + USER-> URL of the repo he impacted

# Observations

* By looking at the plot images, most of the plots show an increased growth rate after a high profile user starts watching. The rate slowly saturates with time
* If a user has particularly high number of followers, the growth rate increases substantially
* If a user has lower than average number of followers(average calculated from the data), the chance of the growth rate being continually increasing is less, showing that the growth rate is mostly independent of his impact
* The growth rates of very popular repos seem not to differ much, even when a high profile user starts watching
* Most of the "social" effect is seen within 1 day of the **event**, similar behavior is also observed during news proliferation in social networking sites like Facebook. Here the workflow is usually like...User watches a repo->His followers get notified; follow the repo->Their followers. So on.
* User do seem to behave in groups, a distinct set of users show high co-incidence, i.e if a user’s starts watching another set users are most probable to follow that repository. The below are set of two users, with their incidence count
  + (('torifat', 'mkol5222'), 642)
  + (('fnu', 'torifat'), 452)
  + (('jasolko', 'fnu'), 342)
  + (('hansstimer', 'payco'), 152)
  + (('rgigger', 'roundhead',32)

**The below are set of three users, with their incidence count**

* + (('anggriawan', 'rgigger', 'jasolko'), 314)
  + (('fnu', 'jasolko', 'torifat'), 134)
  + (('payco', 'roundhead', 'rgigger'), 78)
  + (('fnu', 'hansstimer', 'torifat'), 34)
  + (('rgigger', 'anggriawan', 'payco'), 29)

# Dataset statistics

**Total Repos**: 296456

**Top 10 popular repos**

           Repo\_url                                     watchers      forks       stargazers

1     https://github.com/twitter/bootstrap             28810          0    5623

2     https://github.com/jquery/jquery                16167      16167     2025

3     https://github.com/joyent/node                 14964          0    1794

4     https://github.com/h5bp/html5-boilerplate         14263      14263     2259

5     https://github.com/rails/rails                 14158          0    3164

6     https://github.com/octocat/Spoon-Knife             10590      10590     8838

7     https://github.com/mxcl/homebrew              9602           9602    4358

8     https://github.com/bartaz/impress.js               8594          0    1277

9     https://github.com/documentcloud/backbone    8272          0    1099

10   https://github.com/mrdoob/three.js              7607           7607     972

**Top 10 followed Users**

1             defunkt

2             mojombo

3             torvalds

4             jeresig

5             schacon

6           paulirish

7           ryanb

8             pjhyett

9         visionmedia

10         dhh

**Top 10 events by count**

**eventType         count(eventType)**

1          PushEvent                   140380

2        CreateEvent                42900

3         WatchEvent                29360

4      IssueCommentEvent            20887

5        IssuesEvent                13682

6          ForkEvent                 9967

7          GistEvent                 9082

8       PullRequestEvent             8419

9        FollowEvent                 7592

10       GollumEvent                 4999

**10 fastest growing repositories by number of watchers**

                repo\_url                     FROM  TO

1          https://github.com/mbostock/d3              5346    14027

2          https://github.com/twitter/bootstrap             27717    36352

3          https://github.com/textmate/textmate         1      6532

4             https://github.com/adobe/brackets            29      5935

5          https://github.com/rails/rails                 13986     18009

6      https://github.com/AFNetworking/AFNetworking      2557      6406

7       https://github.com/FortAwesome/Font-Awesome      3333      6442

8             https://github.com/xing/wysihtml5               603      3652

9          https://github.com/HPNeo/gmaps            19      2943

10         https://github.com/ivaynberg/select2            49      2665

**Top 10 Most active repos by event counts**

2      https://github.com/eclipse/eclipse.platform.common       CreateEvent           11903

3          https://github.com/nyarlabo/websites                 PushEvent            4341

4        https://github.com/itroot/reach-github-limit             PushEvent             854

5          <https://github.com/haskell/cabal>                IssueCommentEvent   518

6          https://github.com/rails/rails                      PullRequestEvent         515

7          https://github.com/entoo/portage                     PushEvent             442

8          https://github.com/pulWifi/pulWifi                   IssuesEvent             424

9          https://github.com/mxcl/homebrew                 IssueCommentEvent   406

10        https://github.com/twitter/bootstrap                WatchEvent             373

11          https://github.com/KernCZ/tomato-firmware           CreateEvent             334

# Appendix

## WatchEvent

Watch events happen whenever a user 'watches' a github project

{

"actor": "chinabrant",

"actor\_attributes": {

"email": "sjwu1234@gmail.com",

"gravatar\_id": "e6cf9226e126151a06ed9e8959538089",

"login": "chinabrant",

"type": "User"

},

"created\_at": "2013-04-01T04:01:00-07:00",

"payload": {

"action": "started"

},

"public": true,

"repository": {

"created\_at": "2011-12-24T00:53:30-08:00",

"description": "Trebuchet Launcher",

"fork": false,

"forks": 209,

"has\_downloads": true,

"has\_issues": false,

"has\_wiki": false,

"homepage": "http://cyanogenmod.com",

"id": 3044195,

"language": "Java",

"master\_branch": "ics",

"name": "android\_packages\_apps\_Trebuchet",

"open\_issues": 3,

"organization": "CyanogenMod",

"owner": "CyanogenMod",

"private": false,

"pushed\_at": "2013-03-30T14:27:54-07:00",

"size": 2340,

"stargazers": 145,

"url": "https://github.com/CyanogenMod/android\_packages\_apps\_Trebuchet",

"watchers": 145

},

"type": "WatchEvent",

"url": "https://github.com/CyanogenMod/android\_packages\_apps\_Trebuchet"

}

## FollowEvent

{

"actor": "krishnateja",

"actor\_attributes": {

"blog": "",

"company": "",

"email": "kkteja@gmail.com",

"gravatar\_id": "3778b178758cde9deab511a0b867f08a",

"location": "India",

"login": "krishnateja",

"name": "KrishnaTeja",

"type": "User"

},

"created\_at": "2013-04-01T04:01:10-07:00",

"payload": {

"target": {

"followers": 2,

"gravatar\_id": "fa915cffc1fd9d2dd1fed26225d8ecc6",

"id": 826730,

"login": "cseymourSF",

"repos": 2

}

},

"public": true,

"type": "FollowEvent",

"url": "https://github.com/cseymourSF"

}