GitHub user score calculation

General introduction

GitHub platform attracted the largest community of open-source developers, that support a myriad of projects daily, building a new technological future and changing the landscape of the IT industry. We want to gratify you — the GitHub contributor — for your efforts towards advancing the cutting-edge technologies, and convert your input into a token value on our community platform, based on U°OS blockchain framework.

We quantified your individual GitHub activity, such as commits, releases, pushes and other events in a single score, that is directly converted into the U°OS token units. Tokens can be utilized by the owner as the stake for hardware resources on our network or as a live financial instrument.

Our U°OS Network platform is one of the GitHub habitants, that builds a framework, based on the U°OS blockchain technology, augmented with decentralized U°Community application Together these products allow you to create decentralized communities and accumulate transparent digital reputation within them.

U°OS consensus algorithm rewards network participants for many kinds of social and economic activities they perform, facilitating the network growth and rewarding the user for their input.

We are inviting you to check your score, receive token reward and join U°Community to try out the system, which gives you the opportunity to wrap your personal contribution into a clear reputation profile, own your contribution and turn your effort towards the network development into the network influence and cryptocurrency reward.

Technical introduction

Each GitHub user contributes to the set of repositories, which in their turn have different scales of activity. When calculating your GitHub user score, we account for both the overall number of events within the repository and for your individual activity distribution between several projects. In addition, every event has a different weight (see Table 1).

A joint system of linear equations 5-6 iteratively computes a set of unknown parameters, namely, the scores for repositories and the scores for individual users (8).

Individual user scores, obtained as a result of these calculations, are standardized to sum up to 1 (so they represent a pecentage of the overall score), and are converted into your U°OS token reward, as a share of the token pool,

EVENT	WEIGHT	DESCRIPTION	GitHub Link
MemberEvent	1	user accepts an invitation; removed as collaborator; permissions changed	LINK
PublicEvent	5	private repository is open sourced	LINK
PullRequestReviewCommentEvent	4	a comment on a pull request's unified diff is created, edited, or deleted	LINK
ForkEvent	1	fork repo	LINK
GollumEvent	4	wiki create/update	LINK
ReleaseEvent	5	release is published	LINK
PullRequestEvent	4	anything w a pull request	LINK
IssueCommentEvent	3	issue comment created/edited/ deleted	LINK
PushEvent	3	push to branch	LINK
DeleteEvent	X	delete branch/tag	LINK
CommitCommentEvent	2	commit comment	LINK
WatchEvent	1	star repo. NOT watch	LINK
IssuesEvent	3	anything with an issue	LINK
CreateEvent	2	create a repo/branch/	LINK

Figure 1: Weights of individual events, performed by the users on GitHub

proportionally to that calculated score percentage. Overall token pool fund is 10 million U°OS tokens.

Data

We use the full archive of GitHub events from https://www.gharchive.org, in the format User:Repo:Event. We downloaded 1.3 billion events, spanning a period from January 2016 till February 2019.

Computation

First, we define the weights of GitHub events p_α with α being the event type. We have 14 event types in our database, their weights p_α are predefined by the algorithm (Figure 1).

Second, we define the individual score of a user u with event k in repository r:

$$\hat{W}_{urk} = \sum_{\alpha} p_\alpha I_{urk\alpha} \tag{1}$$

where the indicator function $I_{urk\alpha} = 1$, if the account u contributed to the repository r with k-th event, that has type α, otherwise $I_{urk\alpha} = 0$.

Third, we sum over all k events, submitted by the user u in the repository r, to obtain the score of that user in that repository.

$$W_{ur} = \sum_k \hat{W}_{urk} \tag{2}$$

Then, we standardize individual user and repository scores, respectively, $W_{ur}^{(r)}$ and $W_{ru}^{(u)}$.

$$W_{ur}^{(r)} = \frac{W_{ur}}{\sum_s W_{us}} \tag{3}$$

where the index s enumerates over all repositories in the database.

$$W_{ru}^{(u)} = \frac{W_{ur}}{\sum_v W_{vr}} \tag{4}$$

where the index v enumerates over all GitHub users in the database.

Finally, we obtain:

$$r_r = \sum_u W_{ru}^{(u)} u_u \tag{5}$$

$$u_u = \sum_r W_{ur}^{(r)} r_r \tag{6}$$

Joint system of linear equations in the matrix form:

$$\boldsymbol{u} = \boldsymbol{W^{(r)} W^{(u)} u} \tag{7}$$

And the iterative computing procedure is:

$$\boldsymbol{u^{(n+1)}} = \boldsymbol{W^{(r)} W^{(u)} u^{(n)}} \tag{8}$$

Procedure 8 is implemented, using a standard method of iterations.

The scores that we obtain, as a result of calculation 8, are used to compute Gini coefficient, that measures imcome inequality, in order to control for extreme score inflations (we choose to accept Gini coefficient, that is equal to 0.6). Later, these accepted scores are converted into token share, that is distributed to each user. Users that are not in our database (for example, they have a very fresh GitHub account, so no registered activity) obtain a minimal token share from the pool share, dedicated to this purpose. Users with extremely small scores are truncated from the calculation results and receive minimal share as well.

Outcome of computation

As a result of computation on our dataset, we obtain scores for $2,169,019$ unique GitHub users and also scores for $79,230,475$ repositories.

Meaning of our scores and quick comparison

We gathered a large dataset, that contains information about many types of activities that users perform on GitHub. Comprehensive list of these activities is represented in Figure 1. Unlike scores, obtained in https://gitstar-ranking.com, for example, our scores are computed simultaneoulsy with repository scores, based on a large number of events. This helps to make algorithm more stable and resilient to the work of bots, that might add user rates (stars) in an automated way.

Moreover, star likes given by the users, are corrected, according to the user reputation rate. Therefore, likes assigned by the users with high reputation are not the same as likes given by the users with low reputation.

These two algorithms give different results, meaning that different users will be in the top. Our algorithm is more comprehensive, in the sense that it takes into account many more pieces of information, that contribute to the user reputation score.

Screens of top users

Figure 2 shows top 30 users with their GitHub user ID's, raw scores, token rewards (based on these scores), and also their standardized scores (these standardized scores sum up to 1).

Reputation computation usecase

The work that we did is a very good case example of reputation computation. Users perform valuable activity on some platform, such as GitHub here: these actions can be qunatified and evaluated in many ways, using different algorithms, including ours. Once this reputation is calculated, it can be used by its owner on a single or many platforms. In the case of decentralized blockchain systems, the method of calculation of the reputation remains open and can be changed by the community. This means that the reputation that the user obtains is transparent. In decentralized systems users can receive token rewards, based on their reputation, and influence the network development via voting.

```
23040076        32490.494141    34805.0520171572        0.00348050520171572
29139614        16030.822266    17172.8260094957        0.00171728260094957
27856297        15240.694336    16326.410944692 0.0016326410944692
2354108 14360.630859    15383.6535042401        0.00153836535042401
6697940 13140.786133    14076.9094776016        0.00140769094776016
14790466        6644.550781     7117.89530069084        0.000711789530069084
16239342        6087.720215     6521.3972378954 0.00065213972378954
8655789 6013.096191     6441.45714754842        0.000644145714754842
1341245 5889.576172     6309.13780922755        0.000630913780922755
25385932        5485.649414     5876.43611616309        0.000587643611616309
20407524        5421.823242     5808.06309521501        0.000580806309521501
39814207        5198.140625     5568.44577556968        0.000556844577556968
14175800        5089.804199     5452.39167907243        0.000545239167907243
37936606        5079.839355     5441.71695949883        0.000544171695949883
22429695        4898.768066     5247.74651370909        0.000524774651370909
19940114        4730.67041      5067.67391660878        0.000506767391660878
11215289        4556.795898     4881.41293183105        0.000488141293183105
2662304 4541.070312     4864.56708650919        0.000486456708650919
903479  4450.145508     4767.1649811692 0.00047671649811692
751143  4112.085938     4405.02272295405        0.000440502272295405
13653959        3088.037598     3308.0232255901 0.00033080232255901
30537854        2995.461914     3208.85263485791        0.000320885263485791
30536864        2994.938721     3208.29217063409        0.000320829217063409
30538765        2982.842773     3195.33453147016        0.000319533453147016
53164   2922.086426     3130.25002372059        0.000313025002372059
4170616 2861.389893     3065.22959100085        0.000306522959100084
26384082        2687.73877      2879.20790900907        0.000287920790900907
12962539        2554.740234     2736.73482307824        0.000273673482307824
64996   2443.99292      2618.0980917376 0.00026180980917376
26334818        2440.717773     2614.58962981012        0.000261458962981012
5184102 2435.733398     2609.2501778955 0.00026092501778955
1779249 2352.719727     2520.32277886145        0.000252032277886145
```

Figure 2: Top 30 GitHub users with scores. Column names: GitHub user id, user score, token amount, standardized score (this means %, total sums up to 1)

Code

Code will be released on GitHub by end of May 2019.