Truck Factor

Jingci Wang

February 28, 2019

Introduction

The Truck Factor is defined as the minimum number of team members that have to suddenly disappear from a project before the project stalls due to lack of knowledgeable or competent personnel. [1] This assignment is aimed at calculating the Truck Factors of GitHub repositories as in [1]. The whole process can be split in a number of steps, each described in the following section.

1 Study Design

1.1 Data Acquisition

Systems implemented in the languages mentioned in [1] would be evaluated in this work: JavaScript, Python, Ruby, C, C++ Java, and PHP. Since intuitively a repository with more contributors would be more interesting to the calculation of Truck Factor, repositories with top 150 number of forks would be selected in the following analysis. Retrieving the logs for each repository, commit information are collected from around 1050 repositories. The data collected records name and email of authors and committers in each commit and the timestamps of their actions, the modification information in each commit including the file modified and the number of lines added or deleted, and the message information for each commit.

[1] `https://en.wikipedia.org/wiki/Bus_factor`

1.2 Data Exploration

The information are stored in MySQL database for future reusage. After cleaning the data (formatting the fields, removing unwanted signs like double quotation, converting some fields string to datetime), author name, author date, number of lines added for each file, file path (each file path points to one specific file) and hash for each commit are useful information for finding the truck factor. Furthermore, the scale of file path information is extremely large due to that large amounts of documentations, images or examples are collected which should be discard. Since we only care about the source code files which are the true supports for the repositories, like [1], Linguist library[2] is used to reserve the the wanted source code files. Connecting MySQL with Python, for each repository, three data frames would be selected for following computation: the author and modification information related to this repository, and file list chosen by Linguist. The first two table can be joined by hash value which is unique for each movement in the commit history. Then, use left join with the selected file list to filter the rows associated to uninteresting files to generate the data frame containing information truly relate to truck factor of each repository.

1.3 User Identification

In this paper, the first author of a file is referred to as the one with earliest commit record related with this file. Since several persons may have same names, using the composite of name and email for user identification is considered. However, not all users have provided their email address and it is highly likely that a user owns several email addresses. It is probable safer to only use name to identify users, though keeping the risk of same names (low probability for one repository) and aliases which remains to be fixed in future work.

1.4 Truck Factors

The calculation of the truck factors is the same as in [1] by identifying the Degree of Authorship:

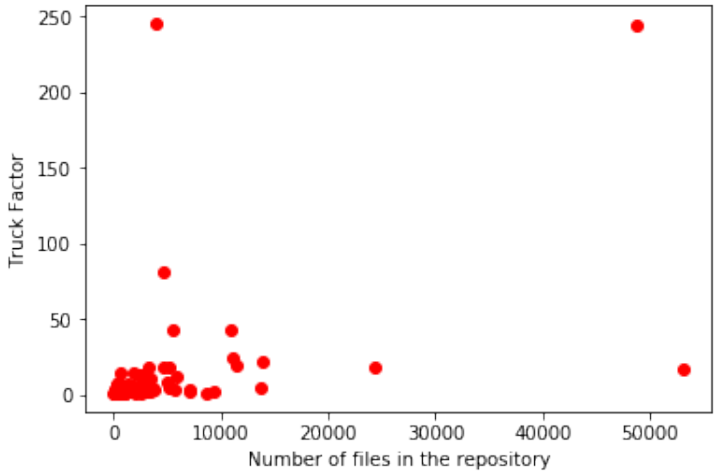
$$DOA = 3.293 + 1.098 \times FA + 0.164 \times DL - 0.321 \times ln(1 + AC)$$

[2] `https://github.com/github/linguist`

Then consecutively remove the author with more authored files in a system until more than 50% of the system's files are without author.

2 Result and Discussion

Due to the computation time, I randomly selected 100 repositories for the calculation of truck factors. As in [1], most systems have a small truck factor. By intuition, the truck factor may correlated with the scale of that repository. However, by drawing the scatter point plot of the number of selected files in the repository and its truck factor, no significant relationship has been found in the small repository sample.



The truck factor results in each language are also listed as **Table** 1-7. Still, no language-specific trend has shown. This may be due to the lack of samples.

On the whole, the truck factors are slightly larger than the results in [1]. Failing to remove aliases may be to blame. Also, some manual work in [1] are not performed in this assignment.

Table 1: Truck Factor results for C++ related systems

TF	Repositories
1	gaoxiang12/slambook
2	assimp/assimp, apache/incubator-weex, hrydgard/ppsspp, flutter/engine
3	gameplay3d/GamePlay
4	FreeCAD/FreeCAD, mysql/mysql-server
5	zcash/zcash
22	llvm-mirror/clang

Table 2: Truck Factor results for C related systems

TF	Repositories
1	antirez/redis
2	kbengine/kbengine
3	greenplum-db/gpdb, emscripten-core/emscripten, reactos/reactos
4	swaywm/sway
5	openwrt/openwrt
6	mruby/mruby, numpy/numpy, videolan/vlc
15	git/git
17	freebsd/freebsd
43	qmk/qmk_firmware
244	torvalds/linux

Table 3: Truck Factor results for Java related systems

TF	Repositories
2	apache/incubator-druid, wix/react-native-navigation, pockethub/PocketHub
3	junit-team/junit4
4	naver/pinpoint
18	bazelbuild/bazel
20	apache/hadoop
43	aosp-mirror/platform_frameworks_base

Table 4: Truck Factor results for JavaScript related systems

TF	Repositories
1	moment/moment, quilljs/quill, webpack/webpack, freeCodeCamp/freeCodeCamp
2	zeit/next.js, chartjs/Chart.js, Modernizr/Modernizr
3	gameplay3d/GamePlay
5	adobe/brackets, emberjs/ember.js
6	angular/material, storybooks/storybook, serverless/serverless
8	angular/angular.js
11	nodejs/node
13	facebook/react-native

Table 5: Truck Factor results for PHP related systems

TF	Repositories
1	nrk/predis, PHPOffice/PhpSpreadsheet, ThinkUpLLC/ThinkUp, WP-API/WP-API, opencart/opencart, phpmyadmin/phpmyadmin, laravel/laravel, phpstan/phpstan, monicahq/monica
2	drupal/drupal, matomo-org/matomo, symfony/debug
3	symfony/routing, cakephp/cakephp
6	yiisoft/yii2
8	PrestaShop/PrestaShop
18	magento/magento2

Table 6: Truck Factor results for Ruby related systems

TF	Repositories
1	sass/sass, samaaron/sonic-pi, rmosolgo/graphql-ruby, jruby/jruby
2	celluloid/celluloid, binarylogic/authlogic, plataformatec/simple_form, hashicorp/vagrant
3	realm/jazzy
4	discourse/discourse
5	ruby/ruby, jekyll/jekyll
6	fastlane/fastlane, chef/chef,
7	diaspora/diaspora
12	rapid7/metasploit-framework
18	rails/rails
24	gitlabhq/gitlabhq
81	Homebrew/homebrew-core
245	Homebrew/homebrew-cask

Table 7: Truck Factor results for Python related systems

TF	Repositories
1	sqlmapproject/sqlmap
2	openai/baselines, Rochester-NRT/RocAlphaGo
3	wagtail/wagtail, mitmproxy/mitmproxy
4	scrapy/scrapy
6	reddit-archive/reddit, home-assistant/home-assistant
7	matplotlib/matplotlib
11	saltstack/salt
13	python/cpython
15	scikit-learn/scikit-learn
18	ansible/ansible

3 Conclusion

The commit information of around 1000 GitHub projects have been collected and imported into database. With the connection of MySQL and Python, truck factor can be calculated using the dataset. However, due to the running time (took over 6 hours for 100 projects), only part of the projects are sampled for calculation. It may be considered to offload all the work to cloud so that more data could be included for further analysis.

References

[1] G. Avelino, M. T. Valente, and A. Hora, "What is the truck factor of popular github applications? a first assessment," PeerJ PrePrints, Tech. Rep., 2015.