Pontifícia Universidade Católica de Minas Gerais

Praça da Liberdade – Noite

Engenharia de Software

Laboratório de experimentação de software

Professor: Jose Laerte Xavier

Alunos: Felipe Fantoni, Mateus Fonseca

Uma Análise Comparativa de Repositórios Python

Introdução

Este relatório tem como objetivo realizar o agrupamento de hipóteses e resultados obtidos sobre as quatro perguntas propostas na atividade denominada Uma Análise Comparativa de Repositórios Python. Os dados contidos neste relatório foram obtidos no dia 08/04/2020.

Metodologia

Para conseguir responder as questões propostas, foi realizada uma consulta para a API GraphQL do GitHub que, através de filtros, retornavam alguns dos dados necessários para a resolução das questões (a consulta era diferente paras as duas fases do trabalho). Portanto, exceto pelo LOC (Lines of Code), com esta consulta obtivemos todos os dados necessários. Como ainda era necessário obter o LOC de cada repositório, foi necessário baixá-los. Para isto, utilizamos a biblioteca GitPython. Com todos os repositórios baixados, adaptamos um algoritmo que realizava a contagem de todos os arquivos .py de todos os repositórios baixados anteriormente. Logo após, com todos os dados disponíveis, o arquivo .csv era alterado para possuir os LOCs dos repositórios e, um último script verificava os repositórios que tiveram problema em seu download ou em seu cálculo de LOCs. Finalmente, com o arquivo .csv montado e com os repositórios com problemas identificados, as análises propostas pelas questões puderam ser realizadas. Estas análises serão descritas ao longo deste documento.

Explicação dos scripts utilizados

Para a resolução deste trabalho, foram desenvolvidos 6 scripts (contando com o main.py):

main.py: Responsável por delegar a chamada correta dos scripts de acordo com a fase da Sprint escolhida no momento de execução.

sprint2_faseX_script.py: Ele é responsável por fazer a requisição com a query GraphQL para a API do github, ele também gera um arquivo texto que mapeia os repositórios e nomes que serão baixados. Finalmente ele gera um arquivo .CSV com todas métricas necessárias, exceto LOC (Lines of Code). Existem duas versões deste script, uma para a Fase 1 e uma para a Fase 2, e suas diferenças estão na estrutura e filtros da consulta para a API.

Util/downloader_e_contador_locs_repositorios.py: Ele é responsável por fazer o download dos repositórios presentes no arquivo texto gerado na primeira etapa e por analisar a quantidade de linhas de código de cada repositório (criando um novo arquivo texto com os LOCs obtidos).

Util/exportador_lista_loc_csv.py: Ele é responsável por recuperar os locs obtidos no script downloader_e_contador_locs_repositorios.py e, analisando o arquivo .csv anteriormente gerado, criar um novo arquivo .csv com a nova coluna de LOC de cada repositório. Este arquivo deleta o arquivo .csv obsoleto anterior.

Util/verificador_repositorios.py: Ele é responsável por analisar o arquivo texto de locs gerado no script downloader_e_contador_locs_repositorios.py. Através desta análise, ele gera um outro arquivo .txt que contém informações (nome e número do repositório na lista) sobre repositórios que não foram baixados corretamente e, consequentemente, não tiveram seus LOCs calculados corretamente. Finalmente, ele também insere neste mesmo texto a quantidade total de repositórios que apresentaram problemas.

Hipóteses

RQ 01. Quais as características dos repositórios Python do Guido van Rossum?

- Acreditamos que os repositórios do Guido van Rossum não serão os mais populares do GitHub, mas devem possuir um número consideravel de estrelas. Acreditamos também, que o valor da mediana de tempo de existencia será grande, tendo em vista que os repositórios pertencem ao criador da linguagem. Supomos também, que o valor da mediana de número de linhas de código será alto, superior a 2.000. Por fim, supomos que o número de forks deve ser elevado, tendo em vista a grande popularidade dos repositórios.

RQ 02. Quais as características dos top-1000 repositórios Python mais populares?

- Supomos que os top 1000 repositórios Python mais populares do Gitub , devem possuir uma idade média elevada, porém não acreditamos que os mais populares são, necessariamente, os mais velhos. Também acreditamos que a quantidade de Watchers será elevada, algum valor a cima de 100. Supomos também, que o valor da mediana do número de linhas de código será elevado, superior a 2000. Por fim, acreditamos que o número de Forks deve ser alto, devido a grande popularidade da linguagem.

RQ 03. Repositórios populares Python são de boa qualidade?

- Presupomos que os repositórios Python sejam de boa qualidade, e que ao comparamos os valores dos Top-1000 repositórios Python com os valores dos repositórios do Guido, eles serão semelhantes.

RQ 04. A popularidade influencia nas características de repositórios Python?

- Depente. Caracteristicas como tempo de vida e número de linhas de código, provavelmente não serão influenciadas, porém, número de estrelas, watchers e forks devem crescer linearmente com a popularidade.

Respostas e Resultados Obtidos

RQ 01. Quais as características dos repositórios Python do Guido van Rossum?

N° de estrelas	N° de watchers	N° de forks	Linhas de código	Idade em dias
35	4,5	2,5	2.858	1.011,5

- Como observado nos valores medianos contidos na tabela acima, as características dos respositórios Python do Guido van Rossum não possuem números tão expressivos. Isso ocorre, pois dos 12 repositórios do criador do Python, apenas um apresentou a grande popularidade que se esperava. Como exemplo, a mediana do número de estrelas é 35, porém o repositório do Guido com maior número de estrelas possui 8180.

- A hipótese se mostrou falsa quanto ao número de estrelas e número de forks.

RQ 02. Quais as características dos top-1000 repositórios Python mais populares?

N° de estrelas	N° de watchers	N° de forks	Linhas de código	Idade em dias
4.309	201	815	6.048*	1.693

- * A quantidade de linhas de código foi calculada com base em 948 repositórios, pois 52 apresentaram erro no momento do cálculo.

- Como observado nos valores medianos contidos na tabela acima, as características dos Top-1000 repositórios Phyton do GitHub são bem expressivas. Todos os valores medianos são altos, especialmente os que tem ligação direta a popularidade do repositório.

- As hipóteses se mostraram verdadeiras.

RQ 03. Repositórios populares Python são de boa qualidade?

Repositório	N° de estrelas	N° de watchers	N° de forks	Linhas de código	Idade em dias
Guido	35	4,5	2,5	2.858	1.011,5
Top-1000	4.309	201	815	6.048*	1.693

- Sim, os repositórios populares de Phyton são de boa qualidade. Ao observamos os valores das medianas contidos na tabela acima, podemos concluir que os repositórios mais populares de Python possuem números muito mais expressivos comparado aos números dos repositórios do Guido. Afim de parâmetro de comparação, o repositório Python mais popular possuiu mais de 80 mil estrelas, enquanto o respositório mais popular do Guido possui pouco mais de 8 mil estrelas. Contudo, é necessário destacar que foi uma comparação de 1000 para 12.

- A hipótese se mostrou falsa.

RQ 04. A popularidade influencia nas características de repositórios Python?

Grupo	N° de estrelas	N° de watchers	N° de forks	Linhas de código	Idade em dias
Top 250	10.520,5	460,5	2.001	6.558	1.677,5
Bottom 250	2.817	123	513,5	3.756,5	1.654

- Sim, definitivamente a popularidade influencia em caracterítiscas dos respositórios. Como observado nos valores medianos da tabela acima, quanto maior a popularidade de um respositório, maior será seu número de estrelas, watchers e forks, porem nem todas as características são influenciadas. A ideia, por exemplo, não tem relação com a popularidade do repositório.

- A hipótese se mostrou falsa quanto ao número de linhas de código.

Link do repositório GitHub deste trabalho

https://github.com/MateusSantosFonseca/Analise-Repositorios-GitHub-Labex