Visualização de Dados – Proposta Trabalho Final

Danilo Ferreira, Guilherme Avelino, Hudson Borges

[1]Departamento de Ciência da Computação, UFMG

Abstract. *Nesse trabalho desenvolveremos um conjunto de visualizações com o intuito de auxiliar na identificação de especialistas de código em sistemas de software. A indentificação do autor/especialista em artefato de código é uma atividade importante para auxilio na manutenção e evolução de software.*

`http://dcc.ufmg.br/~danilofs/dvis/tp1`

1. Introdução

Sistemas de recomendação representam uma tecnologia capaz de auxiliar a atividade natural de identificar especialistas em uma organização. Um sistema de recomendação capaz de fornecer sugestões de pessoas que detenham algum conhecimento sobre um dado artefato de software (arquivo, módulo, etc) é uma ferramenta importante para auxilio na atividade de manutenção e evolução de software [3]. Além disso, sistemas de recomendação de especialistas podem reduzir a sobrecarga sobre as pessoas responsável por identificar tais especialistas e pode prover facilmente alternativas para quando os especialistas conhecidos não estão disponíveis.

Nos últimos anos o GitHub se tornou uma poderosa ferramenta de colaboração para desenvolvimento de software. Atualmente é o maior repositório de código do mundo, com mais de 6.8 milhões de colaboradores e 15.2 milhões de repositórios armazenados. Muito de sua popularidade de se deve a características como controle de versão distribuído e foco em *social coding* [1].

No trabalho aqui proposto, buscamos desenvolver visualizações que auxiliem na identificação de especialistas em artefatos de software. As visualizações aqui propostas serão construídas com objetivo de analisar um conjunto de dados extraídas de sistemas Open Source extraídos do GitHub[1].

2. Abordagem Proposta

Em nosso trabalho, buscaremos, através de informações extraídos de projetos de Software Livre armazenados no GitHub, fornecer visualizações que axiliem a identificar especialistas de código em projetos de desenvolvimento de software.

Nas sub-seções a seguir caracterizamos os dados, definindo critérios para seleção de projetos no GitHub e como será realizada a coleta automática de dados, bem como o tratamento desses para sua visualização.

2.1. Conjunto de Dados

A seleção de projetos armazenados no GitHub se dá não só por sua popularidade, como também pelas facilidades disponibilizadas por tal ambiente, tais como ferramentas de consulta e uma API para extração de informações.

[1]https://github.com/

Os dados a serem utilizados nesse trabalho já estão disponíveis. Os sistemas foram baixados do GitHub e tiveram seus dados tratados durante o desenvolvimento da pesquisa de um dos membros desse trabalho. Detalhes sobre a seleção e tratamento dos dados é fornecida nas seções a seguir.

2.1.1. Seleção de Projetos e Extração dos Dados

A seleção dos sistemas a serem analisados foi baseado em sua popularidade no GitHub, sinalizada através do número de estrelas que o projeto possui. Essas estrelas são atribuídas por usuários do GitHub a projetos que julguem ser de seu interesse. Incialmente foram clonados 600 projetos GitHub, sendo 100 de cada uma das 6 linguagens mais populares no GitHub. Após uma filtragem inicial, que excluiu projetos pequenos (poucos arquivos), com pequeno histórico de desenvolvimento (número de commits), com equipe de desenvolvimento pequena (quantidade de desenvolvedores) e projetos com problema de migração (oriundos de outros tipos de sistemas de gerenciamento de versão) restaram 135 projetos a serem analisados.

As principais características do conjunto de sistemas selecionados é detalhado na Tabela 1.

Tabela 1. Caracterização do Dataset

LANGUAGE	Sistemas	Desenvolvedores	Commits	Arquivos
JavaScript	21	5.656	106.810	23.990
Python	23	8.653	284.719	37.618
Ruby	33	19.960	307.603	33.344
C/C++	19	36.380	1.331.015	155.863
Java	21	4.499	418.003	140.869
PHP	18	3.413	126.896	21.097
TOTAL	**135**	**78.561**	**2.575.046**	**412.781**

Para a identificação de especialistas no código, a principal informação a ser extraída dos projetos é conjunto de *commits*. De cada *commit* são coletadas as seguintes informações: autor do *commit*, arquivos alterados e tipo da modificação realizada. Esses dados são utilizados para o cálculo da autoria de um arquivo, o qual é detalhado na Seção 2.1.2. Para extrair essas informações dos projetos clonados foram utilizados comandos disponibilizados pela ferramenta git, tais como o comando git log.

2.1.2. Tratamento dos Dados

O principal tratamento a ser realizado é o cálculo da autoria de um arquivo, baseada nas informações de *commits* extraídas dos projetos. Para isso utilizamos uma abordagem baseada no trabalho de Fritz e outros [2]. Nessa abordagem o grau de autoria de código (DOA[2]) é definido baseado em três tipos diferentes de autoria de um autor *D*:

- **primeira autoria**, representando se *D* criou a primeira versão do arquivo;

[2] *Degree of Authorship*

- **numero de entregas**, representando mudanças no arquivo, feitas por *D*. Mudancas essas subsequentes a primeira autoria;
- **aceitações**, representando mudanças no elemento não realizadas por *D*.

Utilizando essa esta estratégia para cada par desenvolvedor/arquivo é calculado um valor de DOA. O desenvolvedor com maior valor de DOA para um dado arquivo é considerado seu autor principal. O que precisamos expor nas visualizações a serem realizadas nesse trabalho é como essa autoria é expressada nos sistemas alvo do estudo.

3. Propostas de Visualização

Para auxiliar na análise dos dados, nesse trabalho, planejamos desenvolver duas visualizações principais:

- **Dashboard**: deverá apresentar diferentes visualizações, tais como gráfico de barras e boxplots com objetivo de caracterizar os sistemas estudados com relação ao seu conjunto de desenvolvedores;
- **Distribuição da autoria**: essa visualização tem como objetivo apresentar como é distribuída a autoria dos arquivos dos sistemas em seu conjunto de módulos (representada através de pastas do sistema). Essa é a principal visualização desse trabalho, devendo ser interativa permitindo navegar pela hierarquia de módula/pastas do sistema. Planeja-se utilizar um especialização do TreeMap [3] para essa visualização.

Referências

[1] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, pages 1277–1286, 2012.

[2] T. Fritz, G. C. Murphy, E. Murphy-Hill, J. Ou, and E. Hill. Degree-of-knowledge: Modeling a developer's knowledge of code. *ACM Transactions on Software Engineering and Methodology*, 23(2):1–42, Mar. 2014.

[3] D. W. McDonald and M. S. Ackerman. Expertise recommender. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work - CSCW '00*, volume Philadelph, pages 231–240, New York, New York, USA, 2000. ACM Press.

[3] http://bl.ocks.org/mbostock/4063582