Proposal Report

RStudio Github Workflow Analysis

Juno Chen, Ian Flores, Rayce Rossum, Richie Zitomer

Intro

Git is a Version Control System that was first released in 2005. From 2005 to the present it has grown to be a tool used across multiple domains in different ways. In 2018, one of the main Git online repositories, GitHub.com, contains more than 28 million users. Companies like GitHub recommend workflows such as the Git Flow, however we don't know how many people follow these workflows or others. As well, many users complain that their experiences with Git tend to be troublesome. With this project, we aim to answer two fundamental questions that can enable the development of a new tool that improves and consolidates workflows for users of Version Control Systems. The first question we aim to answer is *What are common subpatterns in the way people use Git?*. With this question we want to see if we can confirm that users follow workflows such as the Git Flow or if they follow other common workflows that are more intuitive for them. The second question we aim to answer is *What are common workflow patterns across Git repositories?*. This question will enable us to understand how different workflows are used in different contexts.

Exploratory Data Analysis

In order to make an easy-to-use alternative to Git, we first need to identify common commit patterns. In order to identify common patterns, we need to be able to generate and then verify hypotheses about what commit patterns are popular among Git users. We can easily generate these hypothesis by visualizing collections of Git commits as graphs. Viewing the raw data in tabular form (where every row is a parent-child connection between commits) is not sufficient for spotting common patterns, as it requires manually searching for each parent's child, and then each child's child, and so on to follow the order. We plan to visualize graph data using NetworkX, a popular Python package that allows for the "study of the structure, dynamics, and functions of complex networks"[1]. Git commits form a directed acyclic graph[2] so we are visualizing them as such. In the future, we would like to be able to extend our work by taking into account and visualizing the relative time between commits.

What are common sub-patterns in the way people use Git?

To answer this question, we are ultimately going to produce a table and a bar chart displaying identified subgraph frequencies in the GitHub dataset. Many fields, including genetic data analysis and social network analysis, deal with analyzing directed graphs, so we studied research papers from those fields to search for possible algorithms. Most of the algorithms are based on Weisfeiler-Lehman Graph Kernels, which iterate over nodes and edges, re-labels and groups them, then represents the features in a vector.

The first approach we found was Node2Vec, which is an algorithmic framework for learning continuous feature representations for nodes in networks. While Node2Vec only takes in the neighbourhood information of the nodes, Sub2Vec improves on it by including the subgraph structure, such as clique, degree, and size of subgraph. With Sub2Vec, we can first embed the subgraphs of each Git commit, and then apply clustering algorithms like k-means on the embeddings.

Another approach is to frame this problem as finding common motifs in a network. In network science, motifs are subgraphs which occur in a network at a much higher frequency than random chance[5]. We plan to identify motifs ourselves either manually (by associating common Git patterns with their motif) or algorithmically (by sampling subgraphs) and then counting their occurences in the network of Git commits.

What are common workflow patterns across Git repositories?

For analyzing and comparing features at a project level, we propose Graph2Vec[6]: A neural embedding framework to learn data-driven distributed representations of arbitrary sized graphs. We propose Graph2Vec over other subgraph analysis algorithms (Node2Vec[3] and Sub2Vec[4]) due to their lack of ability to model global structure similarities, instead focusing on local similarities within confined neighbourhoods. Using Graph2Vec, we can learn the differences within Git projects in an unsupervised manner and use the generated embeddings to cluster similar graphs together with widely-used clustering algorithms.

It may be possible to use Graph2Vec to also identify common sub-patterns. As Graph2Vec learns features across graphs that make them unique, we may be able to extract these features from the model to identify the important sub-patterns.

Analysis Limitations

Our proposed algorithms assume undirected and unweighted graphs. To apply them to our study, we may need to extend the algorithms to work with directed edges (where we can present the Git workflow) and weighted edges (where we can represent the time elapsed between commits).

Conclusion

In this project we aim to answer the two main questions *What are common subpatterns in the way people use Git?* and *What are common workflow patterns across Git repositories?*. To answer these questions, we will be using data science techniques such as graph embeddings and clustering. However, these questions examine Git at a high level and will lead to conclusions of how projects are used at a high level. Extensions of this project include diving into users' local Git usage. With the research knowledge attained from our study, our partner will be able to develop the necessary tools to improve Git.

References

[1] Networkx: https://networkx.github.io/documentation/stable/index.html

[2] Git: https://medium.com/girl-writes-code/git-is-a-directed-acyclic-graph-and-what-the-heck-does-that-mean-b6c8dec6505

[3] Node2vec: https://cs.stanford.edu/people/jure/pubs/node2vec-kdd16.pdf

[4] Sub2vec: https://link.springer.com/chapter/10.1007/978-3-319-93037-4_14

[5] Motifs: https://link.springer.com/chapter/10.1007/978-3-319-16112-9_2

[6] Graph2Vec: https://arxiv.org/abs/1707.05005