Propject Proposal **David Wu, Esteban Rey, Andrew Bassilakis**
CS 221

Introduction

What makes a particular programming project successful? What qualities do we look for in these projects that draws us in? What types of developers do we like to work from?

Github is an online host for open-source projects where anyone can post their code for everyone else to see and use. It harbors a huge number of projects, and is a common platform for developers to share and collaborate. Programmers everywhere rely on software off Github for their own independent or commercial projects. With great popularity comes great variety. Each project is different, has a different style, and has different reach. Some projects are surprisingly prolific, with thousands of 'stars', 'watches', and 'forks'. Some are less so. Given that so many projects exists on the platform, can we predict the success, in terms of popularity, of a given project on Github? If so, what are the relevant features, and how do they contribute to the popularity prediction?

Model - Inputs and Outputs

We will utilize a 3TB dataset from Kaggle and Google BigQuery of all Github. This dataset maintains millions of files, commits history, repos, and other nuggets of data that we will reap in our modeling. However, this Kaggle dataset is not all inclusive, and therefore, we will also rely on web scraping of Github to produce the additional data that we require. We will focus on projects that have exclusively C++ code, to narrow the data into a manageable size as well as better suit feature engineering.

Our model essentially aims to learn some function $f(\phi(x))$ where x is a specific github project, vectorized through some featurization process into $\phi(x)$, and the output of $f(\phi(x))$ is a metric of the success of that project.

There are, in turn, many ways to determine the success of a GitHub project. We could look at the number of 'stars' - a vote of affirmation by user - the number of 'watches' - the act of signing up to be notified of any changes to the project - and the number of 'forks' - the number of other projects that are based off the project. While part of our project will be to fine-tune our definition of 'success', we begin with the following metric for some project x:

$$Success(x) = 0.5 * stars(x) + 0.15 * watches(x) + 0.35 * forks$$

This essentially acts as a weighted average of the number of stars, watches, and forks a projects has. These weights are arbitrary and based off the authors' opinions of the importance of each catagory on overall project success. Part of our project will be to formalize this as we continue.

Our essential goal is to predict success.

Baseline

We implemented a simple baseline using a small section of our dataset and a linear regression classifier. Our feature vector was a simple vector of three features: number of 'commits' and number of pull requests. In our smaller dataset, we had 1057 projects. We trained on 867 projects and classified 290 test projects.

Our resulting squared loss was extremely large, suggesting that number of commits and number of pull requests don't have much of an impact on success. However, if we look just at pull-requests and ignore the number of commits, we get a better average squared loss (622), which is still very high. This suggests that the number of commits does not matter in measuring success, which makes intuitive sense. Pushing bad code many times will not help people use this code. Thus, we have lots of work to do to be able to predict the number of stars. Also, we did train and test this classifier on data that was a random subset of all Github data, so that may have skewed the result. Regardless, we believe that this baseline is a good starting point to perform some extensive feature engineering as well as model testing.

Oracle

Our oracle will be essentially our ground truth. That is, our best case will be 100% accuracy, with no error, on our ground truth for success. It is immediately clear that we probably will not achieve this success. As such, we will design a model such that our oracle for that model will be the success on its training data. That is, our oracle will be the error that is created when we test our resulting model on the training data.

Relevant Concepts and Challenges

We will pull from Machine Learning primarily to deal with our data. We have a already labeled set of data, so we will logically use topics from supervised learning. However, we also have a large amount of data to look at across our set, so we may also use unsupervised learning techniques to better understand our data as the project continues.

Our main challenge will be featurization and developing a model. Featurization will come from relevant understanding of the domain. We will be using our own analysis and opinion to determine what features, both immediately from our dataset and from NLP analysis on the actual code, could be relevant. Abiding by Occam's Razor, we want to limit the number of features, so we we want to naturally make sure our features are all relevant. This will come primarily as an engineering decision for us.

Another interesting challenge will be what sort of model to use. Most likely, our features will not be linearly separable, so most classifiers we have covered in our work so far might not reach a high level of success. Part of our project will be exploring different ML models and weighing their strengths and weaknesses in solving our challenge. For example, a neural net may have very high success, but it may be more difficult to determine how the different features go into the prediction process.

Previous Work Done

Borges, Hora, and Valente predicted the future number of 'stars' and the future Github ranking based off the growth of the number of 'stars' over the past 6 months. [Borges, Hora, and Valente] Jiang et al. have looked more qualitatively into why certain repositories are 'forked' from. They find that increased ability to fix bugs, care for the owners, and the relative success of the creator all matter. [Jiang et al.]