Automatic Project Rater

Hebi Li, James Nhan

October 28, 2016

1 Introduction

Open source projects are popular among developers, because it is free to use and everybody can submit change requests, resulting in high quality of software. There are numerous projects hosted on Github. However, open source projects have different qualities. Projects developed by in-experienced developers tend to have lower quality compared to projects with high reputation. Knowing the quality of a project is beneficial to developers, students interested in learning the project, as well as the potential users. Judging the quality of a project is non-trivial, and may require manual effort and domain knowledge.

This work aims to provide a general prediction model for the quality of projects, based on their textual features.

2 Proposed Approach

First, we collect data as ground truth. Projects can be rated by different metrics: a) download times b) lifetime length c) commit number d) a rating system: e.g. github star e) number of contributors f) number of watchers. Specifically, we propose to mine the meta data from github as our corpus, and use the number of stars as the quality ground truth. We may also use other metrics, such as download times, along or combined together. Considering the huge number of projects on github, we may choose to set to threshold, i.e. only projects with at least certain number of stars will be included into the corpus.

We propose to use mainly the file-level and textural features to predict the quality. The proposed features include: 1) number of tests 2) number of documentation 3) number of source files 4) total source line of code 5) total source line of code in source directory (excluding libraries) 6) average length of source files 7) every depth of the file structure 8) average branching factor of the directories.

Some attributes of a project may affect the effectiveness of these metrics. For example, the language used in the project may affect the rating model. C languages tend to have many stand-alone test files, while Java projects have unit test method mixed into source code. There are also features that are hard to collect the may influence the precision of the ground truth we selected. For example, a lot of javascript projects on github have many stars, partially because they have a dedicated webpage as a demo of their project. It is interesting to see the difference induced by these attributes.

Finally, We plan to use linear classifiers and *Support Vector Machine(SVM)* with non-linear kernels, to learn the relation from features to rating.

3 Research Questions

- Can we predict the quality by the proposed features?
- Is there any features that have strong relation to the quality? Can we still predict quality after removing of such features?
- Does the pattern differs for projects using different programming languages?
- Does the pattern differs for different metrics we use as ground truth
- Can we apply the model to projects to other source, such as sourceforge, google code, and GNU projects.

4 Challenges

- Collecting data from github, non-trivial considering the huge number of projects. By the time of writing this document, more than 3 million projects have at least one star on github.
- Which learning strategy fits this study, in terms of efficiency and effectiveness.
- Discovering more expressive features during research.