StackRank: Obtaining Relevant Technical Forum Posts Based on User Source Code

Fall 2013 CS97 Senior Conference Project Proposal

O'Connor, Melissa `moconno2` Pitser, Mallory `mpitser1` Yang, Jack `yyang4`

October 10, 2013

1 Abstract

As the amount of code freely available on the internet increases, sophisticated tools suited for obtaining relevant code are becoming more in demand. Currently developers have to manually parse through help forums and general search results that may have relevant questions but cite irrelevant code. We aim to present a tool that will identify and rank technical help forum posts to provide the user with more applicable results. We will use a variety of tools, such as StackExchange, the Stack Overflow API, and Moss, a plagiarism detection program. In combination, these will allow us to compare larger code segments uploaded by a user to posts on Stack Overflow in order to provide a ranked list of recommended posts.

2 Motivation

Finding relevant resources for programming questions often proves to be a tedious search through piles of tangentially related code. We aim to assist those people who are searching for answers and guidance in specific programming projects. Currently, someone who is looking for answers to a question may conduct a search through a search engine and then continue to tech-oriented sites such as Stack Overflow to further parse through relevant-seeming answers. One major problem with this approach is that many questions may seem relevant at first glance, but the code samples may reveal different underlying issues. We seek to remedy this problem by providing a more specialized search option in which users' code is directly compared with code found on sites such as Stack Overflow.

3 Background

There are a few different tools that are currently and that have previously existed that discover code available on the web. Google's Code Search, which was designed to help people search for open source code all over the web, was shut down in January 2012. This approach aimed to allow people to find code that matched certain user-provided parameters. Another tool that has become popular for a different reason is Moss[1], a plagiarism detection program that compares a batch of uploaded source code files to each other. Its goal is to find partial matches between code snippets and provide a measure of potential plagiarism.

4 Idea

Our main idea is to parse the code that the user uploads, and compare it with code in various open-source repositories. We intend to apply our approach to Stack Overflow in particular, given both its popularity and reputation among programmers, and our initial observation that it has a well-developed API ('StackExchange') for retrieving individual posts. We will explore the use of Moss (the existing plagiarism detection tool mentioned earlier) to assign a similarity score to each comparison. Thereafter we will rank and display the results using an algorithm of our fashioning. We hope to apply the existing plagiarism detection program in a novel way that can help programmers discover existing usable code.

5 Milestones

5.1 Milestone 1: Learning and Tool Exploration

We will read papers on plagiarism detection techniques to understand how different approaches work. Our research will also include testing open-source services and programs we can use in order to select one that best suits our intentions. Although we are reasonably certain that the StackExchange API provides the functionality we desire, during this phase, we will ensure that this is indeed the case and explore further options if necessary.

5.2 Milestone 2: Development

We will connect the the various building blocks, such as the plagiarism detection program, the repository API, and the basic ranking algorithm to evaluate and present a list of relevant results. Beginning in this phase, we will be building the code upload interface while also developing heuristics to rank the supposedly relevant matched posts. Our initial interfaces and heuristics at this stage will be minimal with the goal being consistent and accurate interaction between the moving parts.

5.3 Milestone 3: Refinement

Finally, we will develop a more sophisticated heuristic that evaluates the various code matchings based on similarity, relevance, and so on. At the same time, we'll build a user-friendly interface for users to upload their code and to view relevant search results.

References

[1] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local algorithms for document fingerprinting. *SIGMOD*, pages 76–85, 2003.