May 1620 Reflection

Looking Back

Initially, StackOverflow seemed like the prime candidate for our social approach to generating specifications. We went with this assumption for most of first semester and did not post questions to there until the latter half to see what our results looked like. By posting sooner, we would have been able to discover that StackOverflow was not suitable for us sooner, and then move on to our new approach. Unfortunately we spent too much time with StackOverflow which ultimately led to limited time for our final approach in the second semester.

After seeing the issues we ran into now, we believe that we could have taken a better approach toward the question and answer idea. Simply developing off of a custom site (such as the one May1639 is building) would have allowed us tighter control over the type of questions and answers we would receive. Obviously this has its own issues as well. Getting new users onto the site could be difficult. We feel that with our combined experience we likely could have developed a StackOverflow clone in a short time which would have allowed us the time to then develop our own questions and userbase.

Another failure we experienced was anyCode. anyCode was and still is a tool that we feel is closely related to our project. We gave up on using it as it took too long for us to get it in a working state, which we feel was the right decision at the time. If we had more time, we might have been able to work with it more to get useful results. Although it was deeply ingrained with java function calls, the techniques they used likely would have proved to be useful to us as we have limited machine learning knowledge, and this tool generalized itself across the java code base.

Our final approach of using StanfordNLP to get information about documentation sentences and then use this to find expressions and create specifications proved to be initially successful. StanfordNLP was a difficult tool to get started with due to limited use and lack of documentation and examples. Once we had figured out most of the components and how the data was stored, we found it to be incredibly useful.

The final results we achieved with our analysis were good for initial results. We we able to successfully create specifications for up to 50% of the java.util package. These results were better than initially anticipated. Some packages were not nearly as good, however. We were much less successful on the java.io package. This is due to much of the documentation describing exceptions related to files. This documentation does not look like an expression and we found it difficult to match. However, we did see many similar phrases showing up there so it might be possible to use our techniques to get decent results there. The package that we worked the most on was util, so with some time we could probably improve our system for each package.

Future Considerations

While we were happy with our initial results, we had to take an approach that gave us the best results in limited time. This required us to build a system that isn't completely general,

and doesn't scale to new data without manual modifications, but did show that our approach might be a viable solution. We created a parse tree matching program to improve our matching rate that ended up being underutilized due to little time left. This program was able to use the part of speech and named entity recognition annotations on tokens in sentences to find common phrases in documentation text. Right now, the tool just prints out what the common tree structures are with these annotations, and it doesn't run in an automated fashion. We believe this tool could be modified to automatically generate patterns to match against for common phrases.

While we have little knowledge of machine learning, we feel that employing some machine learning techniques on this project would have been useful. We did not have time to attempt doing this, however. The major difficulty lies with the lack of training data. Since there are not specifications for a sample set available to train off of, a large amount of specifications would have to be written by hand first to create data for the learning algorithms.