**Related Works**

Prevalence and Harmfulness of Code Clones

There are numerous ongoing and completed studies that illustrate the effect of code clones. According to Baxter et al[1], about 5-10% of large-scale computer programs are clones. Baxter et al. also confirmed this by applying their own clone detection technique - to be discussed in the subsequent sections – to an existing large C production software system. They confirmed the average clone percentage of 12.7% for their subject subsystems. Along with that result, the researchers suggested direct relationship between code clones and maintenance costs. So, if 12.7% of the code clones were found, then maintenance costs can be saved about 12%, assuming even distribution of maintenance cost across source.

Juergens et al.[2] presented how do inconsistent code clones affect software quality, by answering their research question “can inconsistent clones be indicators for faults in real systems?” The researchers analyzed 4 commercial projects and 1 open sources project. As a result, they found 3-23% of the inconsistent clones being actual faults. To this research question, they concluded that inconsistent code clones had statistically more faults than the average code, thus “inconsistencies can be indicators for faults in real systems.”

On the contrary, Rahman et al.[3] argued the above position. They chose 4 open source C projects as their subjects and studied the possible relationship between bugs and code clones. To cut to the point, they could not confirm any strong relationship between bugs and code clones from the data given. Contrary to Juergens’ argument they denied the correlation between inconsistent code clones and faultiness. “clones don’t really small that bad!”

However, the subject projects for above two studies differ greatly in the development language, domain and availability. While the first research focused more on object-oriented languages, such as java and C#, the latter studied only C, which is a sequential language. Since object-oriented languages tend to suffer more from inevitable code clones [4], their discrepancy could be explained by this fact. Also, Juergens incorporated some commercial product into their studies. Perhaps, commercial products – maybe because of the reason inherent from the difference in the development workflow – are more susceptible to code clones.

Code Clone Detection Techniques

This section could be same as the corresponding section from introduction

Problem Solving with Machine Learning

In effort to (even more) automatize knowledge engineering process, machine learning became a promising solution for various fields. According to Langley[5], machine learning actually being used in industries, such as motor pump diagnosis, automatic celestial image classification, etc. The common characteristics of these solutions were that the problem was in known domain (specific task), and that machine learning outperformed the previous solutions. These practical applications made it seem possible to apply machine learning in classifying bugs – and hopefully we can expect some improvement from previous solutions.

Yet, machine learning still has limitation: deprecated learning quality under incomplete information. In order to train the machine better, we need more information that represents the knowledge domain well. Especially, examples from minority class are hard to obtain. Here minority class means a set of data whose occurrence is very rare. Khoshgoftaar et al[6] carried out an empirical study to find out the ideal class distribution in such case. In their conclusion, they noted that 2:1 (majority:minority) showed the best learning quality. Assuming that buggy instances are rarer than the benign one, we could have applied this rule of thumb. In the future, perhaps we can improve the accuracy and recall rate using this principle.

References

1. Baxter, I.D.; Yahin, A.; Moura, L.; Sant'Anna, M.; Bier, L.; , "Clone detection using abstract syntax trees," *Software Maintenance, 1998. Proceedings. International Conference on* , vol., no., pp.368-377, 16-20 Nov 1998  
   doi: 10.1109/ICSM.1998.738528  
   URL: <http://ieeexplore.ieee.org.libproxy.utdallas.edu/stamp/stamp.jsp?tp=&arnumber=738528&isnumber=15947>
2. Juergens, E.; Deissenboeck, F.; Hummel, B.; Wagner, S.; , "Do code clones matter?," *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on* , vol., no., pp.485-495, 16-24 May 2009  
   doi: 10.1109/ICSE.2009.5070547  
   URL: <http://ieeexplore.ieee.org.libproxy.utdallas.edu/stamp/stamp.jsp?tp=&arnumber=5070547&isnumber=5070493>
3. Rahman, F.; Bird, C.; Devanbu, P.; , "Clones: What is that smell?," *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on* , vol., no., pp.72-81, 2-3 May 2010  
   doi: 10.1109/MSR.2010.5463343  
   URL: <http://ieeexplore.ieee.org.libproxy.utdallas.edu/stamp/stamp.jsp?tp=&arnumber=5463343&isnumber=5463276>
4. Sandro Schulze, Sven Apel, and Christian Kästner. 2010. Code clones in feature-oriented software product lines. SIGPLAN Not. 46, 2 (October 2010), 103-112. DOI=10.1145/1942788.1868310 <http://doi.acm.org/10.1145/1942788.1868310>
5. Pat Langley and Herbert A. Simon. 1995. Applications of machine learning and rule induction.Commun. ACM 38, 11 (November 1995), 54-64. DOI=10.1145/219717.219768 <http://doi.acm.org/10.1145/219717.219768>
6. Khoshgoftaar, T.M.; Seiffert, C.; Van Hulse, J.; Napolitano, A.; Folleco, A.; , "Learning with limited minority class data," *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on* , vol., no., pp.348-353, 13-15 Dec. 2007  
   doi: 10.1109/ICMLA.2007.76  
   URL: <http://ieeexplore.ieee.org.libproxy.utdallas.edu/stamp/stamp.jsp?tp=&arnumber=4457255&isnumber=4457184>