Construction of General Software Defect Prediction Model via Machine Learning

Li Jidong 17M38124

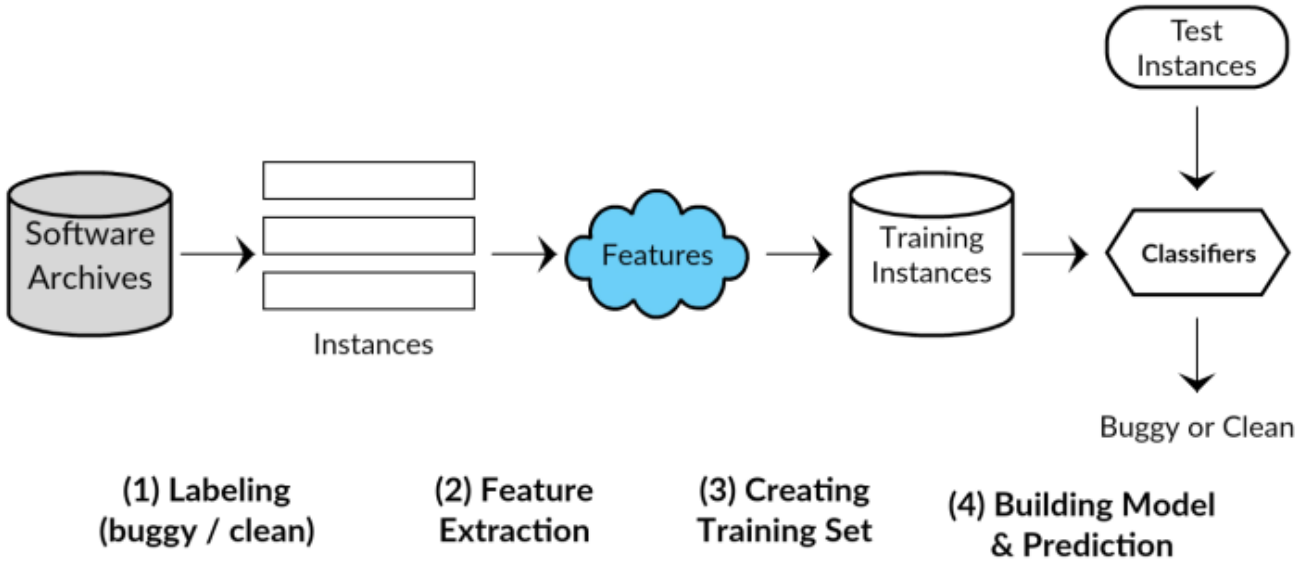
2018/09/13

Background(1)

- Software testing is a serious section
  - Lead to severe problems
- How to test the software efficiently?
  - Software defect prediction(SDP)

Background(2)

- Predict which parts of software may have defects



Related Research(1)

- Metrics
  - Code metrics
    - McCabe metrics[1]
    - Halstead metrics[2]
    - CK metrics[3]
  - Software processing metrics
    - Change metrics[4]
    - Developers based metrics[5]

[1] McCabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,2(4):308-320. [doi: 10.1109/TSE.1976.233837]
[2] Halstead MH. Elements of Software Science (Operating and Programming Systems Series). New York: Elsevier Science Inc., 1977.
[3] Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994,20(6): 476-493. [doi: 10.1109/32.295895]
[4] Nagappan N, Ball T. Use of relative code churn measures to predict system defect density. In: Proc. of the Int'l Conf. on Software Engineering. 2005. 284-292. [doi: 10.1145/1062455.1062514]
[5] Graves TL, Karr AF, Marron JS, Siy H. Predicting fault incidence using software change history. IEEE Trans. on Software Engineering, 2000,26(7):653-661. [doi: 10.1109/32.859533]

Existed problems

- Traditional metrics fail to capture semantic information of programs[6,7]
- Latest machine learning algorithms were not considered
- Have not deeply classified the defects

[6] S. Wang, T. Liu, and L. Tan, “Automatically learning semantic features for defect prediction,” in ICSE’16: Proc. of the International Conference on Software Engineering, 2016.
[7] J. Li, P. He, and MR. Lyu, ”Software defect prediction via convolutional neural network,” in QRS’17: Proc. Of the International Conference on Software Quality, Reliability and Security,2017

Research Purpose

- Construct effective software defect prediction model via machine learning

- Sub-goals:
  - Develop useful metrics
  - Develop a methods to classify buggy
  - Verify whether the latest machine learning algorithms are more superior than traditional one

Approach

- Extract the semantic feature from programs
- Use the machine learning classifier to predict the modules
- Evaluate the performance by AUC,F1 and recall score
- Deeply classify the priority of buggy modules

Conclusion

- Background
  - Efficiency of software testing
- Goal
  - Effective software defect prediction model
- Sub-goals
  - Useful metrics
  - Priority of buggy module
  - Comparison of classifiers

Preparatory Slides

Related Research(1)

- semantic information of program as metric[7,8]
  - Semantic information

```
1. static void myFunc (Queue myQueue) {
2.     int i;
3.     for (i = 0; i < 10; i++) {
4.         // insert i to the tail of the queue
5.         myQueue.add(i);
6.         myQueue.remove();
7.         // remove the head of the queue
8.     }
9. }
```

File1.java

```
1. static void myFunc (Queue myQueue) {
2.     int i;
3.     for (i = 0; i < 10; i++) {
4.         // remove the head of the queue
5.         myQueue.remove();
6.         myQueue.add(i);
7.         // insert i to the tail of the queue
8.     }
9. }
```

File2.java

Fig. 1. A motivating example. *File2.java* will encounter an exception when calls *remove()* at the beginning if the queue is empty.

[8] X. Zhang, K. Ben & J. Zeng, "*Cross-Entropy: A New Metric for Software Defect Prediction*", in QRS's 18: Proc. of: International Conference on Software Quality, Reliability and Security,2108

Related Research(2)

- Workflow