# Using Active Learning with Text Filtering to Generate a Support Vector Machine Training Set

Joseph M. Geyer

23 March 2009

**Abstract**

The need to understand software systems is an important part of their update and maintenance. If one does not understand a software system, he/she will have difficulty modifying, maintaining, or updating it. This can be costly in terms of both time and money. Reverse software engineering alleviates this by creating models of a system to aid system comprehension. A well known problem in this domain is the concept assignment problem [2]. This is the task of assigning human level concepts or meaning to the code that actuates it. This problem can be extended to the class level where the the goal is to assign concepts to classes. Carey and Gannod [5] have automated the classification of the concept classes in a software system by using support vector machines. Support vector machines require a training set to train the learner, however, manually labeling this training set is inefficient. The goal of the proposed research is to present a method and tool to semi-automate the creation of a training set for support vector machine learning in the context of reverse software engineering.

## 1 Introduction

The documentation of the design of legacy software systems is often neglected as the system ages. It is easy for updates and modifications to be made to a software system without those changes reflected in the design models. Knowing the high level architecture of a software system is critical to its continued efficacy. Chikofsky and Cross [6] state that reverse engineering is the process of examining a software systems to extract design knowledge in order to facilitate maintenance and future updates. There exists an increasing demand for the ability to create models and blueprints of the design of software systems.

A well known problem in the domain of reverse software engineering is the concept assignment problem [2]. This is the task of assigning human level concepts to the code that actuates it. Carey and Gannod [5] have developed a tool to automate the classification of the concept classes in a software system by using support vector machines. However, this requires a great amount of work to manually label a training set of classes as either concept or non-concept to train the learner. The goal of the research presented in this paper is to present a method to reduce the load on this aspect of training the support vector machine

# 2 Background and Related Work

## 2.1 Background

In the domain of reverse software engineering, the concept assignment problem is a classic task for recovering design rationale. Another important technique is that of active learning where the user observes the results of the software tool and indicates whether it was correct or not. Finally, in machine learning supervised learning is used successfully in many cases. We are interested in classification area of machine learning where we can classify an item in one of two different classes.

### 2.1.1 Reverse Engineering and the Concept Assignment Problem

In order to maintain or update a software systems, it is important to understand it. Reverse software engineering takes a software systems and extracts knowledge about it's design to further the understanding of the system. Chikofsky and Cross [6] define reverse software engineering as

- identify the system's components and their interrelationships and
- create representations of the system in another form or at a higher level of abstraction

With the number and complexity of legacy software systems, there is a real demand for reverse software engineering.

The famous concept assignment problem presented by Biggerstaff [2] involves two parts. The first is to identify the real human concepts that are in a software system. For example, a concept might be to "calculate revenue". Then the user assigns that concept (calculate revenue) to the code that accomplishes that concept. This was first described on a line-by-line micro level of the code. However the principle still applies to a larger scale when we consider the class level. In Carey and Gannod's [5] work, they consider classes as either concept or non-concept classes. They assigned each class a vector of object oriented metrics and used machine learning to classfiy the concept and non-concept classes. They used supervised learning with cross validation to assess the validity of their classifier.

### 2.1.2 Active Learning

Active learning is where the user can play a part in the acceptance or rejection of the classifications of the learner. This was successfully used in the the work of Bowring et al. [3]

## 2.2 Related Work

Sartipi did some research on reverse engineering with machine learning. This went fairly well, as he was published. Others also published in this area.

# 3 Proposed Research

This section discusses the research to be conducted. It starts by defining the research problem and the the proposed solution. Next it examines the feasibility and validity measures of the proposed solution. This section also includes some preliminary results conducted on a UML Modeling software system. Finally, a timetable for completing the research is presented.

## 3.1 Specific Research Problem

Classification problems with supervised learning algorithms allow the machine to create a learner from the training set that will be applied to the rest of the data set. The user must manually classify a subset of the data so that the learner can automatically classify the rest of the instances in the data set. As an example, if a researcher wants to create a training set that is 10% of a software system with 5000 classes, he/she would have to manually classify 500 classes. This creates a bottleneck in the productivity of the reverse software engineering process. Through this research, we want to reduce the load of manually classifying the training set.

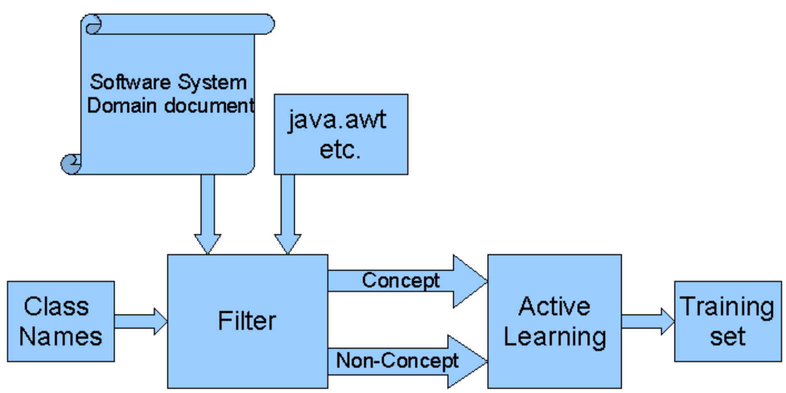


Figure 1: Flowchart Model of Proposed Research

## 3.2 Proposed Solution and Methodology

The proposed solution will be implemented in an eclipse plugin that utilizes filtering and active learning techniques. This plugin will be an extension on the plugin by Carey et al. [5] The flowchart in figure 1 is shows a model of the proposed solution.

### 3.2.1 Training Set Selection via Filtering

We start by using a filtering technique. In our problem, every class is either a *concept class* or *other*. The SVM learner created by Carey [5] requires the user to manually classify at least one positive example and at least one negative example. The goal of this step is to automatically filter out probable concept classes and those classes that are most likely not a concept class. Software systems have a concept domain associated with its high level functioning. Some domain examples are given below.

- Database
- IDE
- Data Mining
- UML Modeling
- Chat Client
- Text Editor
- Online Store
- Chat Client
- Project Manager

In the UML Modeling, some examples of conceptual terms are *inheritance*, *use case*, and *association*. These terms can be taken directly out of a textbook or other document on UML. The same is true of other domains as well. We can extract these terms from any document (user's guides, manuals, textbooks, tutorials, etc.) associated with the concept domain of the system. These keywords together make up the *concept filter*. Likewise there are other terms that are usually associated with the inner workings of a software system. We compiled a list of 98 terms that would filter out classes that are probably not concept classes. These terms are from the java.awt package. Some examples are *JButtonItem*, *Scrollbar*, and *JTextPane*. These non-concept keywords are used in the *non-concept filter*.

The filtering of the classes takes place by examining matches of substrings of the software systems class names with substrings elements of the *concept filter* and the *non-concept filter*. If a match occurs on the *concept filter*, then the class name is placed in a list of probable concept classes. If a match occurs on the *non-concept filter*, then the class name is placed in a list of probable non-concept classes. If a class does not get matched by either filter, then it is not placed in either list. In this way, we have reduced the number of classes to choose from for manual classification.

#### 3.2.2 Training Set Selection via Active Learning

Next we allow the user to accept or reject the automatic filtering. Figure 1 shows the concept and non-concept classes as input into the active learning module. Here the user will be presented with the list of probable concept classes generated by the *concept filter*. The user accepts individual classes as concept by using a checkbox. In accepting a class as a concept class, the user is finalizing the positive examples for the training set for the learner. Similarly, the user will be presented with the list of probable non-concept classes generated by the *non-concept filter*. Again, the user accepts individual classes as non-concept classes by clicking a checkbox. After this process is complete, the training set has been finalized, complete with positive and negative examples. This training set can then be used with the support vector machine to classify all of the classes in the system as either concept or other.

### 3.3 Feasibility

### 3.4 Validation Approach and Measures

TODO: For this section, review some of the statistics used by Carey. Also consider exploring the quality of the training set as a result. For example, maybe the technique for generating a training set falls within the acceptable categories, but is it complete insofar as it describes or defines what a concept class should and should not be?

### 3.5 Preliminary Results

TODO: After cleaning up some of the statistics, this section will contain some of the initial results from the ArgoUML experiments.

Argo UML is a UML modeling software systems. Since the domain of the systems is UML, I used a glossary of UML terms found online. TODO: Find this source. Also put the document as an appendix to this proposal? I parsed through this document to extract all of the terms.

### 3.6 Timetable

## 4 Conclusion

## References


[1] Kristin P. Bennett and Colin Campbell. Support vector machines: hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13, 2000.

[2] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas E. Webster. Program understanding and the concept assignment problem. *Commun. ACM*, 37(5):72–82, 1994.

[3] James F. Bowring, James M. Rehg, and Mary Jean Harrold. Active learning for automatic classification of software behavior. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*, pages 195–205, New York, NY, USA, 2004. ACM.

[4] Gerardo CanforaHarman and Massimiliano Di Penta. New frontiers of reverse engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 326–341, Washington, DC, USA, 2007. IEEE Computer Society.

[5] M.M. Carey and G.C. Gannod. Recovering concepts from source code with automated concept identification. pages 27–36, June 2007.

[6] E.J. Chikofsky and II Cross, J.H. Reverse engineering and design recovery: a taxonomy. *Software, IEEE*, 7(1):13–17, Jan 1990.

[7] Y.-G. Gueheneuc, K. Mens, and R. Wuyts. A comparative framework for design recovery tools. pages 10 pp.–134, March 2006.

[8] K. Sartipi. Software architecture recovery based on pattern matching. pages 293–296, Sept. 2003.

[9] K. Sartipi, K. Kontogiannis, and F. Mavaddat. Architectural design recovery using data mining techniques. pages 129–139, Feb 2000.

[10] Jochen Seemann and Jürgen Wolff von Gudenberg. Pattern-based design recovery of java software. In *SIGSOFT '98/FSE-6: Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 10–16, New York, NY, USA, 1998. ACM.

[11] Jelber Sayyad Shirabad, Timothy C. Lethbridge, and Stan Matwin. Supporting maintenance of legacy software with data mining techniques. In *CASCON '00: Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*, page 11. IBM Press, 2000.

[12] Jinhui Tang, Yan Song, Xian-Sheng Hua, Tao Mei, and Xiuqing Wu. To construct optimal training set for video annotation. In *MULTIMEDIA '06:*

*Proceedings of the 14th annual ACM international conference on Multimedia*, pages 89–92, New York, NY, USA, 2006. ACM.