Chapter VIII

Modeling Relevance Relations Using Machine Learning Techniques

Jelber Sayyad Shirabad, University of Ottawa, Canada
Timothy C. Lethbridge, University of Ottawa, Canada
Stan Matwin, University of Ottawa, Canada

Abstract

This chapter presents the notion of relevance relations, an abstraction to represent relationships between software entities. Relevance relations map tuples of software entities to values that reflect how related they are to each other. Although there are no clear definitions for these relationships, software engineers can typically identify instances of these complex relationships. We show how a classifier can model a relevance relation. We also present the process of creating such models by using data mining and machine learning techniques. In a case study, we applied this process to a large legacy system; our system learned models of a relevance relation that predict whether a change in one file may require a change in another file. Our empirical evaluation shows that the predictive quality of such models
makes them a viable choice for field deployment. We also show how by assigning different misclassification costs such models can be tuned to meet the needs of the user in terms of their precision and recall.

Introduction

Software maintenance is widely recognized to be the most expensive and time-consuming aspect of the software process. To maintain a system properly one needs to have at least a partial understanding of the system; this, in turn, requires knowing about the relationships among elements of the system. As the software ages and grows in size, one would expect to see that an increasing proportion of the relationships become undocumented. In particular, latent relationships among system components are created, and some existing relationships become deprecated, that is, remain present for historical purposes only. In a large, “legacy” system there is no single person who completely understands the system and its network of relationships, yet such systems control many aspects of modern society. It is important to note that while there is no crisp definition for most complex relationships in software systems, in many cases one can identify instances of such relations among some components of the system.

The complexity of a software system is clearly related to the complexity of the network of relationships in the system. A key tenet of our research is that a software engineer who needs to investigate software problems and maintain a software system will benefit from knowing what elements of the system are related to each other. In this chapter, we introduce an abstraction called the relevance relation that is used to represent relationships that would be useful for a maintainer to know. A relevance relation maps a tuple of system elements to a value indicating how related they are. Each tuple in a relevance relation therefore contains a set of related elements.

Well-managed software projects maintain repositories that keep track of software revisions and use mechanisms to record software problem reports as well as the changes applied to the software to address these problems. These software change repositories reflect a history of the system, which includes actions that result in the creation of new relationships and the strengthening of the existing relationships in the software. This history also reflects the collective experience and knowledge of software engineers who have maintained the system.

Conceptually, machine learning systems learn models from past experience that can be applied in future unseen situations or scenarios. Therefore, the main thrust of this chapter is to show how one can use machine learning and data mining algorithms and techniques to discover relevance relations hidden in the historic data available in software.

To better demonstrate these ideas, we will present a case study where we learn a specific relevance relation among source files in a software system. We will derive this relation from maintenance history: files that have been changed together when making previous changes are considered to be relevant to each other for the purposes of future changes. However, in principle, relevance relations can hold among any type of element or component in a software system.
Design and Implementation of a Cognitive Tool to Detect Malicious Images Using the Smart Phone
www.igi-global.com/article/design-and-implementation-of-a-cognitive-tool-to-detect-malicious-images-using-the-smart-phone/127012?camid=4v1a