Chapter 7.22
Integrating Projects from Multiple Open Source Code Forges

Megan Squire
Elon University, USA

ABSTRACT

Much of the data about free, libre, and open source (FLOSS) software development comes from studies of code forges or code repositories used for managing projects. This paper presents a method for integrating data about open source projects by way of matching projects (entities) across multiple code forges. After a review of the relevant literature, a few of the methods are chosen and applied to the FLOSS domain, including a comparison of some simple scoring systems for pairwise project matches. Finally, the paper describes limitations of this approach and recommendations for future work.

INTRODUCTION

Free, libre or open source software (FLOSS) development teams often use centralized code forges, or repositories, to help manage their project code, to provide a place for users to find the product, and to organize the development team. Although many FLOSS projects host their own code repository and tools, many projects use the tools hosted at a third-party web site (such as Sourceforge, ObjectWeb, or Rubyforge). These code forges provide basic project/team management tools, as well as hosted space for the source code downloads, a version control system, bug tracking software, and email mailing lists. There are also directories of FLOSS software (such as Freshmeat and the Free Software Foundation directory) that try to gather into one convenient place material about projects interesting to a particular community.

Much open source software engineering research has been focused on gathering metrics from code repositories. Many aspects of the repository-based software development process have been studied in depth, and repository data collection is important for these studies (see Conklin, 2006 for background). The FLOSSmole project (Howison, Conklin, and Crowston, 2005) was created to consolidate metadata and analyses from some of these repositories and directories into a centralized collaboratory for use by researchers in industry and academia. As of this writing, FLOSSmole includes data and analyses from Sourceforge, Freshmeat, Rubyforge, ObjectWeb, Debian project, and the Free Software Foundation (FSF) directory of free software. One of the challenges mentioned in Conklin (2006) in creating this kind of collaboratory is in integrating the data from these various sources. It seems reasonable that a project might be listed on several directories *and* have a listing on a code forge. However, sometimes a project will be listed in multiple forges too, usually because the project has migrated from one forge to another over time, or because the project wishes to "grab" the unique namespace for its project on a certain forge so it will register at that forge without an intention to ever actually use that space.

In any case, when integrating project data from multiple sources, we must first identify which project pairs are matches. In other words, we want to find out which projects are listed on multiple forges. For example, is the *octopus* project on ObjectWeb the same as the *octopus* project on Sourceforge or the project also called *octopus* on Freshmeat? If we can devise a scoring system for determining whether a project pair is a match, then can we automate the matching process?

The focus of this article is entity matching (and duplicate identification) for this kind of data integration, as applied to the domain of FLOSS projects. Section 2 outlines some terminology from the study of data integration problems and gives a background of entity matching algorithms. Section 3 describes the FLOSS domain in terms of entities and duplicates. Section 4 gives an example of applying some of the algorithms for entity matching to this domain. Section 5 outlines limitations of this work and gives recommendations for future study.

ABOUT ENTITY MATCHING

The act of integrating multiple data sets and finding the resulting duplicate records ("matches") is nearly as old as database processing itself. In practice and in the literature, this set of processes is known by many names (Bitton and DeWitt, 1983; Hernandez and Stolfo, 1985; Winkler, 1999; Garcia-Molina, 2006): merge/purge, object identification, object matching, object consolidation, record linkage, entity matching, entity resolution, reference reconciliation, deduplication, duplicate identification, and name disambiguation. The term *entity matching* will be used in this article.

Within the larger activity of data integration, the act of matching entities is not to be confused with the act of schema reconciliation. Schema reconciliation refers to the act of matching up columns or views in different data sources, and using data or metadata to make the match. For a trivial example, suppose a field in Table A is called *url* but it is called *home_page* in Table B. To resolve these schemas, the analyst could create a global schema or view that encapsulates both underlying schemas. This task can be done manually, or can be automated through various machine learning techniques (such as Batini and DeWitt, 1986; Doan, Domingos and Halevy, 2001; Rahm and Bernstein, 2001). Schema reconciliation and entity matching are related, but not identical, tasks of data integration. Most often the schema reconciliation will happen first, followed by the "merge" task, and finally by the eventual "purge" of duplicate data. However, if data sources are kept separate throughout the matching process, then the act of schema reconciliation could include a "merge" between disparate entities.

Agree/Disagree and Frequency-Based Matching

The simplest form of entity matching is what we will call the agree/disagree method: take two