Using FLOSS Project Metadata in the Undergraduate Classroom

Megan Squire[1] and Shannon Duvall[2]
1 Elon University, Elon, NC, USA megan@elon.edu,
WWW home page: http://facstaff.elon.edu/mconklin
2 Elon University, Elon, NC, USA sduvall2@elon.edu,
WWW home page: http://facstaff.elon.edu/sduvall2

Abstract. This paper describes our efforts to use the large amounts of data available from public repositories of free, libre, and open source software (FLOSS) in our undergraduate classrooms to teach concepts that would have previously been taught using other types of data from other sources.

1 Introduction

This brief paper describes our efforts to use the large amounts of data available from public repositories of free, libre, and open source software in our undergraduate classrooms to teach concepts that would have previously been taught using other types of data from other sources.

We will first present the brief history of the FLOSSmole community data repository, and then we will outline the ways in which we have used this publicly-available data about open source software in our classes to teach standard computer science and information systems concepts. We conclude with recommendations for future work in the area of applying FLOSS data in the undergraduate classroom.

2 Background of FLOSS Data

The courses and applications described in this paper use data provided by the FLOSSmole[1] project [4]. FLOSSmole is a collection of datasets about free, libre, and open source software (FLOSS) development. This project was started in 2004 and is still active, with approximately 30 gigabytes of data added each month. The FLOSSmole project team acquires most of the data in the collection, but the project also accepts data donations from related project teams, and works collaboratively with teams that collect related data. Most of the data in FLOSSmole now is metadata describing FLOSS development projects. FLOSS development teams often use centralized code repositories, sometimes called forges, to help manage the programming and communication efforts for each software project. Examples of

[1] FLOSSmole. http://ossmole.sf.net

public code forges are Sourceforge[2], Rubyforge[3], or Google Code[4]. In addition to providing a centralized place for finding and downloading FLOSS software, code forges provide services such as mailing lists for discussion among programmers, version control systems for releasing software, and bug tracking for defects and feature requests. There are also numerous directories of FLOSS software (such as Freshmeat[5] and the Free Software Foundation[6] directories) that attempt to index FLOSS projects and basic metadata about these projects.

Repositories and directories provide a valuable resource for the software user community to find and download relevant software, but they also provide a wealth of data that can help researchers answer questions about the current state of FLOSS development: how many projects are there, how many people have downloaded the software, what are the programming languages or operating systems that are most popular, how are the teams organized, what types of problems are being solved with open source software, how effective is open source as a development methodology?

FLOSSmole currently collects data from Sourceforge, Freshmeat, Rubyforge, ObjectWeb[7], and the Free Software Foundation, as well as data from the Debian[8] project (Debian is a Linux distribution that includes thousands of packages of FLOSS software). Data is collected from the public web sites of these forges/directories on a monthly or bimonthly basis, depending on the forge in question, and this data is re-released through the FLOSSmole web site in a variety of formats: downloadable text files (tab-delimited for import into Excel or other software), downloadable SQL files (CREATE and INSERT statements suitable for import into a favorite database application), and through a live query tool (free registration required to obtain a username and password).

The data available in the FLOSSmole project is varied and plentiful. There are over 200GB of data in the database now, all of it available for download or live query. Each dataset is given a unique number indicating which forge the data came from, and date on which the data was collected. There are currently 150 datasets in FLOSSmole, spanning over four years. It is possible to track the evolution of a project over time by identifying the name of a project and tracing it through the four years of datasets. It is also possible to analyze the rate of growth in new projects and the death or waning of existing projects. The next section will address the specifics of how this database of FLOSS projects can be used to meet various course goals.

[2] Sourceforge. http://www.sf.net
[3] Rubyforge. http://rubyforge.org
[4] Google Code Hosting. http://code.google.com/hosting
[5] Freshmeat. http://freshmeat.net
[6] Free Software Foundation Directory. http://directory.fsf.org
[7] ObjectWeb. http://www.objectweb.org
[8] Debian Project. http://www.debian.org

3 Courses and Goals

There are two main ways that this FLOSS metadata has been used in our courses here at Elon University: in undergraduate coursework (specifically in the database administration course), and in undergraduate research experiences (students conducting individual research projects with faculty mentors).

Other researchers have written about the use of open source software in undergraduate classes, mostly in reference to the cost savings of using software that doesn't require payment [6], of the benefits of having students work on projects that they may find particularly relevant or advantageous in terms of "real-world" applications [5, 7], or which might be interesting from an open source culture point of view [1]. Here we focus on the ready application of a large (and growing), conveniently located, heterogeneous data set that happens to be about open source software development. As such, using the data set effectively in classes will necessarily involve some measure of curiosity on the students' part about the open source software development culture and the history of this movement.

3.1 Database Administration

This course is a typical, one-semester introduction to database management and administration targeted to sophomore-junior level students. The course covers design of databases, the relational model, structured query language (SQL), and basic administration of database systems. With a typical enrollment of 20 students, the course is taught in a computer lab, and serves as an introduction to design and implementation in both Oracle and MySQL environments. FLOSSmole data is used in two main ways in this course: first, because the FLOSSmole datasets are so large, the students have an opportunity to "play with" an extremely large amount of data, and second, students are given the chance to design new solutions to data integration issues. We will describe both of these avenues of inquiry here.

3.1.1 Querying Large Data Sets

To learn how to construct queries in SQL, students in the database administration course are typically given "toy" datasets to manipulate. These datasets can come with a textbook or are given from the instructor. Typical textbook databases [2,8] may include a health care application, an online electronic commerce site for a company, or a university student-course management system. As the students learn design skills, they are typically asked to use these sample databases until they are able to create their own databases and populate these databases with sample data. In some cases, students may learn to load data into their databases with the help of XML files or bulk upload routines. These are all useful skills and do reflect real-world tasks and real-world applications. However, students remain unchallenged in contemplating how to approach a database of this size. Students with experiences consisting solely of manipulating sample databases that can be easily downloaded or that can fit on a CD-ROM remain unconvinced about the need for "efficient" queries. Students who

have dutifully learned that it is right and proper to normalize their databases may be resistant to the idea of de-normalizing. Exposure to ultra-large databases such as FLOSSmole can make these lessons more effective.

We found that motivating students to consider efficiency in their queries and designs required databases with dozens of tables, each with millions - and in some cases billions - of rows. Running queries in this sort of environment is a wholly different animal than running queries in a sample textbook database. Indexing becomes necessary, not just a nicety. Using query optimization techniques is required. Students learn how to kill long queries and how to change server parameters to limit their exposure to the "bad" queries of other students. Most important, students are able to comprehend the difference between an un-normalized database and a de-normalized database. The intentional de-normalization of a database is a strange idea to students at first, but they quickly see the need to reduce query-processing time. The concepts behind data marts and data warehousing now flow naturally as obvious solutions to problems inherent in large data stores.

Yet, we can make the argument that any extremely large dataset could have presented similar work environment for students. In addition to our bias in favor of using a dataset with which we are so familiar, the FLOSSmole dataset has some distinct advantages over other large datasets. First, it is growing at a rapid rate. This ensures that students will have slightly different experiences with the data from month to month and from semester to semester, which is good for keeping the course fresh and relevant. Second, working with FLOSS data gives ample opportunity for the instructor to field questions and provoke discussion about the philosophy of free, libre, and open source projects. Students might not get this exposure in other courses that use proprietary technologies. Finally, there are numerous open questions about open source software development that these students can actually participate in answering. Because FLOSSmole is a community resource, many of the researchers who use FLOSSmole data discuss their findings on the public mailing lists and on the project wikis. This means that students can try their hand at answering real research questions using the data in the FLOSSmole system, or they can replicate findings of other researchers.

Appendix A of this paper lists several questions that students in this course (spring 2007 and spring 2008) came up with to answer about open source software development using the FLOSSmole data. Students were first briefed on the goals of the FLOSSmole project, and they reviewed with the instructor the type of data that was included in the project database. They then brainstormed lists of "interesting questions" in small groups. These questions were aggregated by the instructor, and then the students were sent to a local version of the FLOSSmole database (created using the publicly-available data dumps) to write the SQL queries that would answer these questions.

3.1.2 Data Integration

One of the open questions that FLOSSmole researchers grapple with is determining whether a given project in Repository A is the same as a project listed in Repository B. There is, of course, no limit on the number of repositories or directories with

which a project is listed. This means it can be challenging to determine which projects are the same across these many repositories and directories. For example, is the project called *octopus* on Sourceforge the same as a project called *octopus* on Freshmeat? How can we be sure? With hundreds of thousands of projects and dozens of code forges, it is next to impossible to manually match all projects. Common project names, a highly mobile developer base, and ever-changing web addresses exacerbate this problem. In practice and in the database literature this broad class of data integration problems is called entity matching.

After working with the FLOSSmole database for the semester, and reading background work on this subject [10], students in the database course were able to propose various heuristics as solutions to this problem. For instance, they recognized that FLOSS projects lacked a good unique identifier, but they proposed that there might be a series of other attributes that could be used to identify the projects instead. For example, most students recognized that some combination of web address, project name, descriptive keywords, and other attributes like programming languages or license type could be used to ascertain a match. Students then proposed various weighting mechanisms for the confidence they felt in using these other attributes. Finally, students were then able to implement data storage solutions (including ERD and SQL) for these newly weighted "possible match pairs" in their own MySQL instances.

After students became familiar with the data in the FLOSSmole database, they were able to answer questions such as the ones shown in Appendix A in this paper. Questions 3 and 5 in the Appendix are especially relevant to the entity matching problem since they deal with cross-matching projects from different forges using the attributes of those projects. A similar set of questions that more directly address the entity matching problem is shown in Appendix B.

3.2 Undergraduate Research, I

At our institution we are fortunate that there is ample opportunity for undergraduate research with faculty mentors. One student has been working with the FLOSSmole project as a developer for two academic years. He joined the project after first being introduced to FLOSSmole data in the database administration course described above in his second year at Elon. The main contribution of this student was to write and maintain a framework application for the data collection tasks at the largest forge, Sourceforge. The student used the Python programming language and the background knowledge gained from a separate parallel processing and distributed computing course to implement a distributed job queueing system. This student has subsequently expanded the job queueing system to handle additional important forges (such as Eclipse[9] and Debian) and to handle additional tasks, such as collection of public mailing list data. He is currently working to expand the job

[9] Eclipse.org. http://eclipse.org

queueing system to run on a grid architecture with workload management features, such as Sun Grid Engine.

3.3 Undergraduate Research, II

Another student in her third year of study has been involved in the FLOSSmole project from the perspective of studying information architecture. She has been documenting the database schema and creating a wiki for end-users to understand what data is available and how to better interact with the FLOSSmole database. As part of this project, this student has used skills from previous courses including database administration (described above) and web development. She has also begun a long-term analysis of community-based data archives from other fields, such as biology [3], to compare what features from these other resources could be used in FLOSSmole.

3.4 Undergraduate Research, III

Finally, a fourth-year student has been working more extensively on the entity matching problem described in 3.1.2 using techniques learned from our Artificial Intelligence course (further discussed in 4.1.2 below). In particular, he is using the ID3 algorithm [9] to automatically learn a decision tree for deciding if two entities are a match. The original algorithm uses attribute-value pairs (such as "Name fields match - yes, no, or somewhat") and results in a decision tree which makes a binary decision of whether or not the entities match, optimizing the tree to utilize the fewest attributes possible. The student is extending the ID3 algorithm to incorporate a probabilistic "belief" measure indicating how sure the value for the attribute is. The algorithm builds the tree not according to minimizing number of attributes but to maximizing the belief measure of the outcome. The new algorithm better fits the problem of entity matching where values are not rigidly defined.

4 Results and Future Work

We would like to be able to extend our work using FLOSSmole to other areas of the Computing Sciences curriculum. Two curricular areas that seem the most promising are the introductory management information systems course and an upper-division course on artificial intelligence. Both of these courses would benefit from using the FLOSSmole data to provide the same sort of relevance and challenge that students in the database administration courses have experienced. In some cases the work performed by students in the undergraduate research projects described in Section 3 could be precursors to generally-applicable assignments for these other classes.

4.1.1 Management Information Systems

In the MIS course (typically a course taught to first- or second-year students in information systems and the School of Business), students investigate the application of information systems to solve business problems and to make business decisions. Systems development and the build-versus-buy decision are obviously parts of such a course, and so open source software is already part of any current and updated MIS curriculum. It would be interesting to have students study the FLOSSmole data and business-friendly reports about open source such as the Open Business Readiness Rating[10] [11] to understand the issues around using FLOSS in business. The OpenBRR is an attempt to standardize the ratings of open source products as "ready" for use in business. The OpenBRR uses and extends FLOSSmole data to make some of its conclusions about the relative maturity levels of various FLOSS products.

4.1.2 Artificial Intelligence

The Artificial Intelligence course teaches concepts such as intelligent search, machine learning, probabilistic reasoning, and natural language processing, and the FLOSSmole data can be used as a real-world application for these techniques. Section 3.4 describes one student's use of probabilistic reasoning and machine learning techniques to study the entity matching problem. Another possible project is the automatic collection of new data straight from the individual websites of the FLOSS projects. Students can use natural language processing techniques to "read" the sites and parse out the desired data that augments the project data already in the forge. As for the database class, using the FLOSSmole data would motivate using efficient and intelligent algorithms for dealing with large amounts of real-world data.

5 Concluding Remarks

Our use of FLOSSmole data in several undergraduate courses was initially spurred by our university's mission to focus on engaged learning for undergraduate students, and by its adoption of a teacher-scholar model[11] for achieving greater integration between a faculty member's teaching and research interests. We found that using real-world data sets and real-world business problems brings disciplinary problems into sharper focus for students, and we felt that if the data sets had further integration with our own research interests, this would be even more relevant and interesting to the students. What we found was that in the regular classroom, the use of the FLOSSmole data sets did provide a low-stakes but interesting problem domain for students to practice their skills. In the undergraduate research milieu, FLOSSmole

[10] Open Business Readiness Rating. http://openbrr.org
[11] Elon Teacher Scholar Model. http://www.elon.edu/e-web/academics/teacsch.xhtml

problems were of a reasonable size and difficulty level for our students to tackle effectively. With minimal background coursework, students were able to contribute new features to the project, or were able to provide effective solutions to real research problems.

6 References

1. Dionisio, J. D., Dickson, C. L., August, S. E., Dorin, P. M., and Toal, R. (2007). An open source software culture in the undergraduate computer science curriculum. *SIGCSE Bulletin.* 39, 2 (Jun. 2007), 70-74.
2. Elmasri, R. and Navathe, S.B. (2004). *Fundamentals of Database Systems (4 ed)*. Addison-Wesley.
3. Howe, D., Costanzo, M., Fey, P., Gojobori, T., Hannick, L., Hide, W., Hill, D.P., Kania, R., Schaeffer, M., St. Pierre, S., Twigger, S., White, O., & Rhee, S.Y. (2008). Big data: The future of biocuration. *Nature*, 455. 47-50, 4 September, 2008.
4. Howison, J., Conklin, M., & Crowston, K. (2006). FLOSSmole: A collaborative repository for FLOSS research data and analyses. *International Journal of Information Technology and Web Engineering*, 1(3), 17–26.
5. Meneely, A., Williams, L., and Gehringer, E. F. (2008). ROSE: a repository of education-friendly open-source projects. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (Madrid, Spain, June 30 - July 02, 2008). ITiCSE '08. ACM, New York, NY, 7-11.
6. O'Hara, K. J. and Kay, J. S. 2003. Open source software and computer science education. Journal of Computing Sciences in Colleges. 18, 3 (Feb. 2003), 1-7.
7. Patterson, D. A. (2006). Computer science education in the 21st century. *Communications of the ACM* 49, 3 (Mar. 2006), 27-30.
8. Ramakrishnan, R. and Gehrke, J. (2002). *Database Management Systems (3 ed)*. McGraw-Hill.
9. Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach (2 ed).* Prentice Hall.
10. Squire, M. (2009). Integrating projects from multiple open source code forges. *International Journal of Open Source Software and Processes*, 1(1), 46-57. January-March, 2009.
11. Wasserman, A., Pal, M., and C. Chan (2006). The Business Readiness Rating Model: an Evaluation Framework for Open Source. In *Proceedings of the EFOSS Workshop*, June 8-10, 2006. Como, Italy.

Appendix A

Here are five sample queries for students in the *Database Administration* course to work out using the FLOSSmole database. The goals of this course are discussed in this paper in section 3.1.1.

1. What are the most popular programming languages used on Sourceforge? In your query, list most popular first, least popular last. Construct a graph or chart to display these values visually.
2. Gather the data to show the change over time in using the terms ‘free’ and ‘open’ to name new projects on Sourceforge. Construct a line chart showing the relative popularity of these terms over time.
3. How many projects on each forge share a name with a project on another forge? In other words, how many projects on Freshmeat share a name with a project on Rubyforge? How many Rubyforge projects share a name with a project on Sourceforge? Write queries to determine these values, then design a diagram to show all pair-wise forge comparisons and their numbers of shared project names.
4. Compare team sizes on Rubyforge, Sourceforge, and ObjectWeb. Each development project has a size, from single-person projects all the way up to hundreds of developers working on a project. Collect the number of projects in each size category for each of these forges, then show these various sizes on a graph for each forge.
5. Write the code to determine the following, then construct a table summarizing the answers: number of Sourceforge (SF) projects listing at least one programming language, number of SF projects listing zero languages, number of SF projects listing at least one operating system, number of SF projects listing zero operating systems, number of SF projects listing a license type, number of projects listing zero license types. Does your table look similar to Table 2 in [10]? Why might your table look different?

Appendix B

Here is a question that prepares students to discuss the concept of *entity matching*, described in this paper in section 3.1.2.

1. Using the FLOSSmole data mart ("small collection of data") on the Elon database server, write the SQL that would generate the information shown in the figures below. In other words, what sort of queries would you have to run in order to be able to have the data to draw a graphic like the ones in Figure 1 and Figure 2 below?

The students do not know this at the time they are given the question, but these figures are taken directly from [10]. After working with this data, the students are given this research paper to read and discuss. In a subsequent assignment, the students are challenged come up with alternatives to the entity matching heuristics given in this paper.

Matching projects by URL has two possible: projects listed on different forges might both display the same external URL, or projects on one forge might actually list the project site on a competing forge as the home page of record. The diagram shown in Figure 1 depicts each forge/directory in FLOSSmole and how many of its projects list another forge as the actual hosting home page. For example, in the diagram, the topmost arrow shows 11 projects on the FSF that actually have Rubyforge listed as the home page. The arrow notation is used to show a direction of the relationship (e.g. 10,044 Freshmeat projects show a home page on Sourceforge, but only 4 Sourceforge projects list a Freshmeat home page). Pairs of forges with no URLs in common are not shown. (No Rubyforge projects list ObjectWeb URLs, and vice versa. Also, as is befitting its status as a directory and not a repository, the FSF directory is not listed as the home page of any projects from the other repositories, so these empty relationships are not shown in the diagram.)

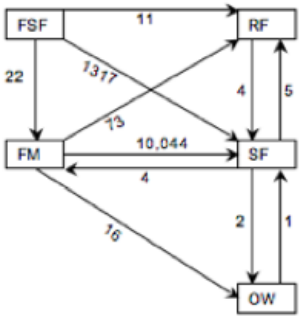

Figure 1: Number of projects at each repository that list a home page at another repository

Most forges require projects to have a unique name (sometimes called the "unixname") within that forge. For example, once a project called *starfish* has been added to Sourceforge, another one cannot be added with the same short unixname. However, multiple projects can have the same "display name"; Sourceforge projects *starfish* and *xstarfish* both have the display name of "starfish". On Sourceforge, 44,112 (39%) of projects have unixnames that are different from their display names (December 2006 FLOSSmole data). Note that the FSF directory has only a requirement for case-sensitive uniqueness in project names. The FSF lists project pages for both *ANT* (telephony application) and *ant* (build tool). There are 54 such ambiguously named projects listed on FSF.

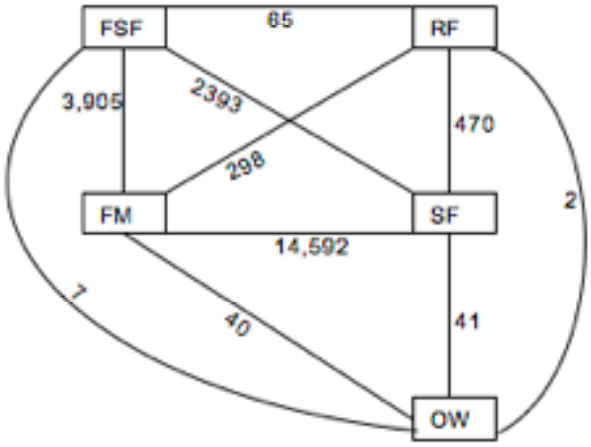

Figure 2: Number of projects at each repository that share an identical short project name