OSSGrab: Software Repositories and App Store Mining Tool

Normi Sham Awang Abu Bakar and Iqram Mahmud

Abstract—One of the main challenges for empirical researchers is to collect software data. However, with the emergence of the open source repositories, they have a large amount of software to choose from for mostly any types of research. Moreover, the mining software repositories research area can be further extended to the mobile apps mining. Generally, searching through large software repositories to look for specific systems can be a daunting task. Therefore, there is a need to build a tool which can expedite and ease the search process so that the researchers can focus on analyzing the data. Our paper presents a tool, OSSGrab, which can be used to automate the search process in the SourceForge software repository, as well as searching through the Android app store. As a result, this tool has managed to save tremendous amount of time that need to be spent for data collection.

Index Terms—App store, mining software repositories, open source, tool.

I. INTRODUCTION

The field of data mining has grown into an extensive network of research which spans many areas of research, including empirical, market, behavior, social and scientific researches. In a simple term, data mining refers to extracting or mining knowledge from large amounts of data. In a broader view, data mining is the process of discovering interesting knowledge from large amount of data stored in databases, data warehouses or other information repositories [1].

In particular, the area of empirical software engineering collects and analyses large amount of data from various sources. Thus, it requires some amount of automation in the data collection and analysis processes. This is where data mining techniques come into the picture. Empirical researchers depend mostly on the data which are publicly available in various repositories. Software repositories contain a wealth of information about software projects. Using the information stored in these repositories, practitioners can depend less on their intuition and experience, and depend more on historical and field data. Examples of software repositories are [2]:

- Historical repositories: Such as source control repositories, bug repositories, and archived communications record several information about the evolution and progress of a project.
- Run-time repositories such as deployment logs contain information about the execution and the usage of an application at a single or multiple deployment sites.
- Code repositories such as Sourceforge.net and Google code contain the source code of various applications developed by several developers.

The popularity of open source systems (OSS) has made it possible to have easy access on the empirical data for research. Researchers now have access to rich repositories for large projects used by thousands of users and developed by hundreds of developers over extended periods of time. This has catalyzed many breakthrough results in many areas of software engineering research, such as software maintenance, metrics and measurement, code quality, developers’ communication, development culture and many more.

The OSS repositories such as SourceForge [3], GitHub [4] and GoogleCode [5] provide a mechanism for developers and users, as well as sponsors to interact and exchange ideas on how to improve the systems. The vast number of systems in the OSS repositories makes it difficult to extract the data in a non-automated way without the assistance of any repository mining tools. Hence, there is a need for a tool to automate the process of mining the systems to be included in the research.

In this paper, we present an open source repository mining tool known as OSS Repository Grabber (OSSGrab) to facilitate the process of mining data from OSS repositories, especially for researchers. This tool manages to save tremendous amount of time which normally spent to collect research data, and the extra time can be spent for data analysis instead.

In addition, the emergence of a variety of applications in the mobile app stores has gained interest among users to search and download the applications. The term “App Store Repository Mining” is becoming more relevant in today’s trend of connectivity among mobile users. In order to ease the mining of these mobile apps, we have included the app mining feature in our tool.

This paper mainly focuses on the discussion of how our tool, OSSGrab, perform the search in OSS repository, especially in SourceForge, including extracting data from the Android App Store.

The remainder of this paper is organized as follows: Section II reviews related work, Section III explains the background of this work. Section IV discusses the Search Techniques in OSSGrab while Section V presents the results/output produced by the tool and Section VI concludes this paper.
II. RELATED WORK

A comprehensive line of work has been reported by several researchers in the area of software repositories mining. In particular, a recent publication by Shang et al. [6] reports on the usage of a web-scale platform known as Pig as a data preparation language to aid large-scale Mining Software Repositories (MSR) studies. They validate the use of this web platform to prepare data for further analysis.

In a similar line, Kiess et al. [7] present a software repository data exchange format based on the Web Ontology Language, EvoOnt which includes software, release, and bug-related information. In addition, they also introduce a Semantic Web query engine called iSPARQL, which can be used together with EvoOnt to perform the software repository mining process. A paper by Voinea and Telea [8] presents a MSR tool, known as CVSgrab, which can be used to acquire the data and interactively visualize the evolution of large software projects.

A recent publication by Harman et al. [9] discusses another form of software repository mining, which they call the “App Store Repository Mining”. They use data mining techniques to extract feature information which later was combined with more readily available information to analyze apps’ technical, customer and business aspects. They applied their tool to collect data in the Blackberry app store. Based on their experience, we were able to add an additional feature in our tool to not only be able to mine the data from OSS repositories, but also be able to collect data from Android app store.

III. BACKGROUND

In the past, the data collection process in the empirical software engineering field was hindered by the difficulty of getting data from software companies. After the advent of open source systems, researchers have the freedom to select the systems to be included in their research and the data collection process now become more convenient to them.

The OSS repositories provide the facilities for users, developers and researchers to interact and improve the quality of the systems. However, the vast number of systems in the repositories makes the process of selecting the relevant systems very time-consuming, thus, there is a need to create a data mining tool which can automate the system selection and at the same time, save the time spent on data collection.

The OSSGrab tool was developed to aid us in automating the data collection process and as a result, instead of spending days manually exploring the repositories to look for the most appropriate systems, we are able to get the results within minutes (depending on the number of systems that match your search criteria and the network speed).

Furthermore, another feature was added to this tool, which can be utilized to collect apps from Android apps store. This can benefit researchers who are interested in studying the trends in app store downloads, correlation between variables in the mobile apps research and many more. Both the repositories and apps mining features are going to be described in greater detail in the next section.

IV. PARSING TECHNIQUES

Our application automates the process of collecting datasets from software and application repositories that has been made public via the World Wide Web. In order to automate the data collection we have developed a program written in Python programming language employing the best pattern recognition algorithms and existing user interface libraries.

A. Parsing Techniques

The parsing techniques are shown in Fig. 1. The application receives a query from the user that specifies the criteria to search along with the repository. The query is then passed to the web-crawler engine that starts crawling the pages from the respective online repository’s API. After loading the pages web-crawler engine hands it down to the parsing engine, which then retrieves the queried data from the mass of text. Once the parsing is done the program writes the collected data in HTML and CSV format for research use. CSV format allows the user to further manipulate the data using rich functions of spreadsheets. Java scripts are added in the HTML to make the data more interactive and useful.

B. User Interface

For preparing the user interface we used PyQt [10], a Python binding of the cross-platform GUI toolkit Qt developed by Nokia. This allows our program to run seamlessly in different operating systems. We have tested our program extensively in Ubuntu, a popular Linux distribution and Windows 7. Technically it should be able to run in Mac OS X. Essentially, users have two main options to choose from, one is to search for systems in the OSS repository, in our case, we choose SourceForge. The other choice is to search for Android apps store. The former is shown in Fig. 2 and Fig. 3, while the latter is illustrated in Fig. 4 and Fig. 5.

Fig. 2 exhibits a simple search, where users need to specify the name of the system they want to look for. The parser will search through the SourceForge repository and will return the result to the users.

Fig. 3 shows the advanced search option where users can select systems based on Categories, Programming Language,
Development Status and Number of Downloads. The Number of Pages keyword means that the users can choose the number of systems that will be displayed on the results page, if the users choose bigger number of pages, the search time will be longer.

The Android app store simple search is illustrated in Fig. 4, where users can enter the app they want to download and then click on the download link. The Android app advanced search is shown in Fig. 5. The users can select the apps using two main keywords, the category and price. The results of the search is further discussed in the next section.

C. Parsing Algorithms

For parsing purpose, the algorithm that we heavily used is known as Regular Expression [11]. It is one of the most convenient algorithms for searching for a pattern in a given text. Instead of looking for an exact text matching it looks for a matching that suffices the pattern. For example:

\(<a \text{href}="/\text{directory/language:java/}\text{jav}\text{a}" />jav\text{a}</a>\)

This is the pattern that is associated with how languages are mentioned in SourceForge. The following regular expression pattern looks for all the languages that are mentioned in that page.

\(<a \text{href}="/\text{directory/language:}[^/]+/\text{\textbackslash{}}/>\text{(S+)}<a>\text{}</a>\)

During matching, this expression is converted into a non-deterministic finite automata (NFA). After that, NFA matches the input string and proceeds to see if it is possible to reach a state where we can claim a successful match. In our program, we used the implementation of Regular Expression that comes as a package with Python version 2.7.2.

We also used BeautifulSoup [12], a python library for parsing HTML documents in the cases where a distinct pattern was not possible to write. It creates a parse tree for parsed pages that can be used to extract data from HTML. Parsing with Regular Expression becomes extremely complicated when data are written in HTML with nested tags.

Another part of the algorithm explores the parsing techniques to mine data from the Android app store. The example of the code snippet is given in Fig. 6. The Android app store contains hundreds of thousands of applications, both free and paid, and the parser needs to find the apps based...
on the categories, as being specified in the Android repository.

VI. TOOL RESULTS

In order to make the HTML output interactive and allow users to sort data according to different variables (i.e. sort by number of downloads, last update etc.), we used TableKit [13], a JQuery library. The CSV output can be used with both MS Excel and Openoffice/Libreoffice.

The HTML output of the search parser in SourceForge is exhibited in Fig. 7, while Fig. 8 shows the output of the Android apps search. The outputs were generated in both CSV and HTML format. The users can sort the output based on the header of the column. The column Download Link will connect the users to the system download in SourceForge. This will provide fast access to the system and the users can directly download the system. From the research point of view, this facility will provide the researchers many options of systems to choose from. In empirical software engineering, researchers need to find as many data as possible, especially when they want to build prediction models, to ensure that the models can be more generalized to the population at large.

Moreover, data collected from the App store can be used in various ways. For example, Harman et al. [9] investigate the correlation between features, ranking and price of the apps. This is interesting where the developers can use the results to...
determine which features to consider when designing apps.

VII. CONCLUSION AND FUTURE WORK

Software repository mining can aid researchers to automatically collect data from the vast amount of systems in the repositories. The usage of the OSSGrab tool can potentially assist researchers to find the data they need in their work, at least it can reduce the time spent for data collection, and they can put more focus on data analysis, instead. In addition, this tool is able to grab the apps in the Android app store and the results of the search can be applied to many areas of research.

Future work would include the cross-repositories search especially in the open source repositories domain. The commonalities between these repositories will be identified and will be utilized to achieve the goal. In addition, the app store mining will be further explored to include more variables and also to investigate potential correlations between the variables.

REFERENCES


Normi Sham Awang Abu Bakar is an assistant professor in the Department of Computer Science, International Islamic University Malaysia. She obtained her PhD in Computer Science at the Australian National University. Her research interests are in the area of empirical software engineering, open source quality, agile methodology, mining software repositories and software engineering education.

Iqram Mahmud is a final year undergraduate student majoring in Computer Science in KICT, International Islamic University Malaysia. He has been developing data mining tools in Python for research purposes since 2011. Iqram has been a member of U. Dhaka and IIUM teams to World Finals of ACM International Collegiate Programming Contest in 2009 and 2012 as a contestant.