Web-Service for Finding Cloned Files using b-Bit Minwise Hashing

Kaoru Ito, Takashi Ishio, and Katsuro Inoue
Graduate School of Information Science and Technology
Osaka University, Osaka, Japan 565–0871
{ito-k, ishio, inoue}@ist.osaka-u.ac.jp

***Abstract*—Source code reuse is a common practice in software development. Since industrial developers may accidentally reuse source files developed by open source software, clone detection tools are used to detect open source files in their closed source project. To execute a clone detection, developers need a database of existing open source software. While a web-service providing clone detection using a centralized database is likely useful, industrial developers are not allowed to submit their source code to a public server on the Internet. To solve the problem, we employ b-bit minwise hashing technique that enables to estimate similarity of documents using only hash values of the documents. Using the method, we implemented a file-clone detection web service; it takes as input a hash value of a source file and returns a list of similar source files in existing open source software. Our hash comparison method is efficient, although an estimated similarity may have a margin of error.**

I. Introduction

Many software developers reuse open source software as a part of their own projects in order to improve development efficiency and quality. However, reuse information is often unrecorded or easily lost through the development history[1].

Code clone detection technique is useful to recover the reuse information. Developers would employ a code clone detection tool to identify clones between their source files and existing OSS files, and recover the reuse information. Based on this idea, we have developed tools named Ichi Tracker[2] and FC-Finder[3]. However, keeping the OSS database always updated is tedious task for individual developers, so we would like to have a third-party WEB service which centralizes the database and its maintenance, and each developer sends local files as query to the WEB service. On the other hand, a developer in a company who develop proprietary software may not be able to upload source files of their projects to such a WEB server, because of a security policy of the organization.

Software Heritage project[4] provides a simple web-service to takes as input a SHA-1 file hash and reports exactly the same file in OSS projects. The service is promising but unable to detect a file if a query file has been modified from the original version. To address the problem, we propose a web-service named Cloned File Detector that takes as input a signature of a query file instead of the full content of the file. We employ b-bit minwise hashing[5] that estimates similarity between two documents using signatures created by minwise hash functions[6]. Company's developers can translate their source files into hash signatures and submit them to the web-service to find similar files in OSS projects. While the web-service can compute similarity values between OSS files and the submitted signatures, the web-service is hard to restore the original file contents from the signatures.

II. File-Clone Detection Web Service

Our web-service named Cloned File Detector is to detect files in a database that are similar to a query file q. Our similarity of source files is Jaccard index of trigrams, i.e. an approximation of string matching [7], defined as follows.

$$sim(f_1, f_2) = \frac{|trigrams(f_1) \cap trigrams(f_2)|}{|trigrams(f_1) \cup trigrams(f_2)|}$$

where $trigrams(f)$ is a multiset of trigrams extracted from a file f. An element of a trigram is a token in source code. A token sequence is extracted using a lexer that ignores comments and white space.

To enable industrial developers to use their closed source file as a query, we employ an approximation using b-bit minwise hashing signature [5]. Conceptually, the technique approximates a similarity of files by using a hamming distance of $b \times k$ bit vectors whose each b bit are corresponding to a trigram in a file. In case of our implementation, we chose parameters $b = 1, k = 2048$; in other words, 2048 trigrams are selected from a file as samples and then translated into a 2048-bit file signature. Given a file signature s_q instead of its original file content q, an estimated similarity of a query q and a file f in a database is computed as follows:

$$sim_e(s_q, s_f) = 1 - \frac{d(s_q, s_f)}{k} \times 2$$

where s_f is a file signature of f and $d(s_q, s_f)$ is a hamming distance of the signatures, respectively. Specially, we treat sim_e as 1 when both SHA-1 hash values are equal and as 0 when sim_e is negative. It should be noted that an estimated similarity may have a margin of error. Our preliminary analysis shows that an absolute error is less than 0.05 for 99.9% of file pair samples.

Our web-service takes as input a file signature s_q and extracts files that are likely similar to q using estimated similarity as follows:

$$CFD(s_q) = \{f | sim_e(s_q, s_f) \geq th\}$$

where th is a threshold (0.7 is used in our current implementation). The extracted files are sorted by their estimated

TABLE I
A LIST OF FILES SIMILAR TO A QUERY FILE `PNGWRITE.C` IN FIREFOX 45.0B5.

Rank	File SHA1	SHA1 w/o space and comment	File Name	Estimated Similarity	Actual Similarity
1	c4419962...	304cd806...	firefox-45.0b5/media/libpng/pngwrite.c	1.000000	1.000000
2	3c27631a...	304cd806...	firefox-47.0b5/media/libpng/pngwrite.c	1.000000	1.000000
3	4410037a...	3fee91a3...	thunderbird-44.0b1/mozilla/media/libpng/pngwrite.c	0.999023	0.999026
4	f1faf25c...	0e3bdb81...	qtbase-opensource-src-5.6.0/src/3rdparty/libpng/pngwrite.c	0.977539	0.974437
5	d5d6ff67...	0e3bdb81...	libpng-1.6.21/pngwrite.c texlive-bin-2015.20160213.39691/libs/libpng/libpng-src/pngwrite.c	0.977539	0.974190
6	96b86f7d...	e47a2fbd...	stella-4.7/src/libpng/pngwrite.c	0.977539	0.974190
7	bc1ad57e...	3f63477c...	libpng-1.6.22/pngwrite.c	0.914062	0.913829

similarity. If tied, an alphabetical order of file paths in OSS packages is used. If a user provided a SHA-1 file hash for q, the web-service highlights exactly the same file in a resultant list.

Our database is based on a snapshot of the Snapshot Archive of Debian GNU/Linux[8]. The archive includes all the existing source code packages released for Debian from 2005 until the present. While Debian package maintainers somtimes apply their own patches, we included only original source tarballs whose names matched a pattern "`*.orig.*`".

Our current implementation supports Java and C/C++ files, while various programming languages are used by OSS projects. We use ANTLR lexers to translate each source file into tokens, and apply a common set of hash functions to compute a file signature. Hence, the service is easy to extend for other languages. We do not discuss hash function implementations due to the limited space.

This is a kind of detection technique for type 3 clones, so that the Cloned File Detector can not extract only type 2 clones. However, the result set includes not only type 3 clones but also type 2 clones.

III. EXAMPLE USE CASE

Our demonstration version of the web service is publicly available[1]. Due to the limited server resource, the service uses a partial database including 573,000 files. A signature computation tool is also available on the website. Given a file signature, the web service returns a list of files that are likely similar to the given file.

Table I shows an example output of Cloned File Detector; the query file is `pngwrite.c` in Firefox 45.0b5, that is a variant of libpng 1.6.21 that supports Animated PNG. The web-service detected similar files in the OSS database and reported their SHA-1 file hash, SHA-1 computed excluding whitespace and comments, file names, and estimated similarity values. A number of file names are linked to a single file SHA-1 because the files have the same content. In the output, the top file is exactly the same as the query file. The second file has the same content except for comments. The third and fourth files are likely different versions of the same file. The fifth file is the original version of the query file, according to a commit message in the source code repository of Firefox.

[1]http://sel.ist.osaka-u.ac.jp/webapps/ClonedFileDetector/

The actual similarity values in the table are not directly provided by the web service, because the web service cannot compute actual similarity without a query file content. The web-service provides a download link for each reported file so that a user can analyze them without uploading a query file. Unless the exact copy of a query file is found in the database, the web-service is hard to know the actual content of the query file.

IV. CURRENT STATUS AND FUTURE WORK

Cloned File Detector is a web-service that takes as input a file signature of a query file and detects similar files in a database. Although estimated similarities may have a margin of errors, users are not required to submit their source code.

In future work, we need to evaluate the accuracy of the estimated similarity compared with the actual similarity. We also would like to extend the service to accept a set of files as a query so that a user can easily investigate an original version of the files.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Numbers JP25220003, JP26280021, and JP15H02683.

REFERENCES

[1] Pei Xia, Makoto Matsushita, Norihiro Yoshida, Katsuro Inoue, *Studying Reuse of Out-dated Third-party Code in Open Source Projects*, Computer Software, Vol.30, No.4, pp.98-104

[2] Katsuro Inoue, Yusuke Sasaki, Pei Xia, Yuki Manabe, *Where Does This Code Come from and Where Does It Go? - Integrated Code History Tracker for Open Source Systems* -, Proceedings of 34th International Conference on Software Engineering, pp.331-341, Zurich, Switzerland, 2012-05.

[3] Yusuke Sasaki, Tetsuo Yamamoto, Yasuhiro Hayase, Katsuro Inoue, *Finding File Clones in FreeBSD Ports Collection*, Proceedings of the 2010 7th IEEE Working Conference on Mining Software Repositories, pp.102-105

[4] Roberto Di Cosmo, Stefano Zacchiroli, *Software Heritage* https://www.softwareheritage.org/

[5] Ping Li, Arnd Christian Konig, *b-Bit Minwise Hashing*, Proceedings of the 19th International Conference on World Wide Web, pp.671–680, 2010.

[6] Moses S. Chariker, *Similarity Estimation Techniques from Rounding Algorithms*, Proceeding STOC '02 Proceedings of the thiry-fourth annual ACM symposium on Theory of computing Pages 380-388

[7] E. Ukkonen, *Approximate string-matching with q-grams and maximal matches*, Theoretical Computer Science, vol. 92, no. 1, pp. 191–211, January 1992.

[8] Debian GNU/Linux, *The snapshot archive*, http://snapshot.debian.org/ (Accessed August 19, 2016).