Evaluation of Source Code Copy Detection Methods on FreeBSD

Hung-Fu Aaron Chang and Audris Mockus

audris@avaya.com

Avaya Labs Research

Basking Ridge, NJ 07920

http://mockus.org/

Motivation and Research Goals

- A key premise of open source is that the code can be used in other projects
  - Reduces risks of project's code being no longer available or supported
  - Provides social value by encouraging innovation (no need to reimplement existing functionality)
- Measure the extent of code reuse
  - Do it on a large-scale
  - On entire sequence not only on a single (often final) version
  - Better validation process and systematic way to understand open source projects

Previous and Related Work

- Filename Comparison method: detect copied files by finding directories that have a large fraction of identical filenames
  - Find directory pairs with a large fraction of identical filenames
  - Consider files with the same names in an identical directory pair to be reused files
  - Applied on Avaya codebase (XXX files) with known instances of copy and found .. .. ..
- Different from CCFinder, ..., in that it looks at entire files that are copied

Objective

- Validate Filename Comparison method
- Ultimate objective is to create a more promising solution to detect the code copy patterns
  - In the large-scale data such as the set of all open source projects
  - Reuse by comparing file content, including each version

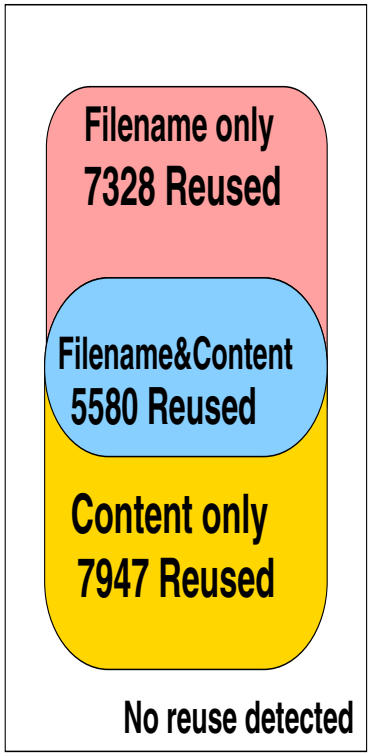
Two level detection

- At filename level
  - Filename Comparison
- At version content level
  - Find the minimal distance between versions of file A and file B using one of the following methods
    - Identical Content: if at least one nonempty version in file A matches some version of file B (No any data process on file content)
    - Nilsimsa (trigram): hash the trigrams that are accumulated from the file content into 64 digit hex code; then compare the number of bytes that differ
    - Vector-Space: build term-by-document matrices and compute the similarity (cosines) between two version contents
    - Abstract Syntax Tree (AST): approximate AST by extracting

control flow keywords and block delimiters; compare the resulting strings using string similarity compari

Filename Comparison Validation

- For Nilsimsa, Vector-Space and AST methods
  1. Apply the three methods on Filename only subset.
  2. Extract and categorize files detected as reused by a single method (in addition to the filename method).
  3. Use random sampling on those sets detected by a single method to select 20 files.
  4. Two experts verify reuse.



Detection Method Comparison

Filename comparison	Very fast on large-scale data. Cannot match versions. Would miss cases where individual files were copied/renamed
Identical Content	Simple. Would miss reuse cases where copy involved a slightest edit or without version history.
Nilsimsa (tri-gram)	Compare files (versions) without cleaning the content. Requires a lot of computation to compare 64 digit hex codes. May suffer from many false positives.
Vector-Space Similarity	Need clean the content (ex: comments). Computation needed to get tokens from files (versions). May suffer from many false positives.
Abstract Syntax Tree (AST)	Can detect control flow reuse. Computation needed to approximate AST from files (versions). Need to know about program language structures.

Identical Content method

Extract content of all versions of all files and place into an associative array indexed by the content ....

FreeBSD

- Has 492583 versions of 57128 files
- Has 360877 unique Contents
- Size of all Contents is 8.16 G
- Average content size is16.6 Kb and standard deviation is 45.4 Kb
- 1.365 filenames on average for each unique content. Standard deviation is 1.3 filenames.

Comparison

- To compare all the methods, we only look at C files
  - Total: 47559 files
- Filename Comparison: 12908 Reused Files
  - Identical Content: 13077 Reused Files
  - Filename only: 7328 files
  - Filename & Content: 5580 files
  - Only Content: 7497 files
- Total number Reused Files is 43 % ((7328+5580+7947) / 47559 = 43%)
- Nilsimsa, Vector-Space and AST methods on Filename & Content cases

Method	Number of reused files detected
AST	3027
Nilisimsa	3143
Vector-Space	1120

Comparison

- Many Reused Files may be missed by Identical Content method due to minor content changes (including comments).
- Reused Files missed by Identical Content method in Filename & Content Zone
- Two experts evaluation results
  - Each method 20 samples
  - Nilsimsa: match primarily on the copyright notice
  - Vector-Space: may be not particularly suited for copy detection
  - AST: Non-Reused Files were not C-language code

Method	Both True	Both False	Disagreement
AST	12	8	0
Nilisimsa	12	7	1
Vector-Space	3	5	12

Conclusions and Future Work

- Copy detection method
- Multiple versions
- Validation process on FreeBSD
- Filename Comparison
  - Easy-to-apply and reasonable method
  - Reused Files in large-scale scope data
    - 60% of Reused Files
    - 4% false-positive rate
- Presenting an analysis procedure to detect and validate file copy patterns in a much larger database of all open source projects
- Computational challenge on Content-based in larger scale data

Acknowledgements

Ahmed Hassan (AST approximation code)

Experimental approach

- Sample a large set of open source projects
- Identify and quantify instances of large-scale reuse
  - not a copy and paste in an editor
  - not a case of reuse where another project is reused as-is through libraries without copying the code
- Identify common patterns of reuse
- Quantify quality and other properties of the reused code

Sample selection and retrieval

- Sample
  - Important projects: Apache, Gnome, KDE, Mozilla, OpenSolaris, Postgres, and W3C
  - Large distributions: Fedora 6, Gentoo, Slackware, FreeBSD, NetBSD, and OpenBSD
  - Development portals: Savannah, SourceForge, and Tigris
  - Random or language specific: FreshMeat, CPAN, RpmForge, and Gallery of Free Software Packages
- Retrieval
  - SVN/CVS, wget, and page scraping (FreshMeat)
  - $13.2M$ files from $49.9K$ bundles
  - $5.3M$ source code files and $38.7K$ bundles after normalization (removing package versions, binary files, ...)

Quantify large-scale reuse

- Method
  - Identify pairs of directories with a large fraction of filenames that are shared between them [**?**] as reused directories
  - Consider files with the same names in reused directories to be reused
- Measures
  - Overall reuse — a fraction of files that are in more than one project
  - Component reuse — a number of projects in which the component is present

Validity

- Sampling process to increase the representativeness of project sample
- The definition of large-scale reuse
  - not a copy and paste in an editor
  - not a case of reuse where another project is reused as-is through libraries without copying the code
- No substantial changes to filenames or directory structure
- The instances of reuse are underestimated (no cases of mistaken identification of reuse were found)