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: linking change histories

- Change history provides a basis for understanding software development and for numerous measurement, reporting, and productivity tools
- Most projects have incomplete change history, e.g.,
  - moved from CVS to Subversion
  - branched from another project
  - were copied (reused) from other projects
- Linking such incomplete histories can help:
  - better quantify the impact of code reuse
  - understand spread of innovation
  - trace software authorship and software-based social networks

Objective: link histories by identifying reuse

- Approach
  - Obtain file-to-file links using the simplest method (Filename Comparison)
  - Identify matching version using more expensive (computationally) method
  - Validate by comparing with other methods that are likely to have different errors

Context: Filename Comparison Method

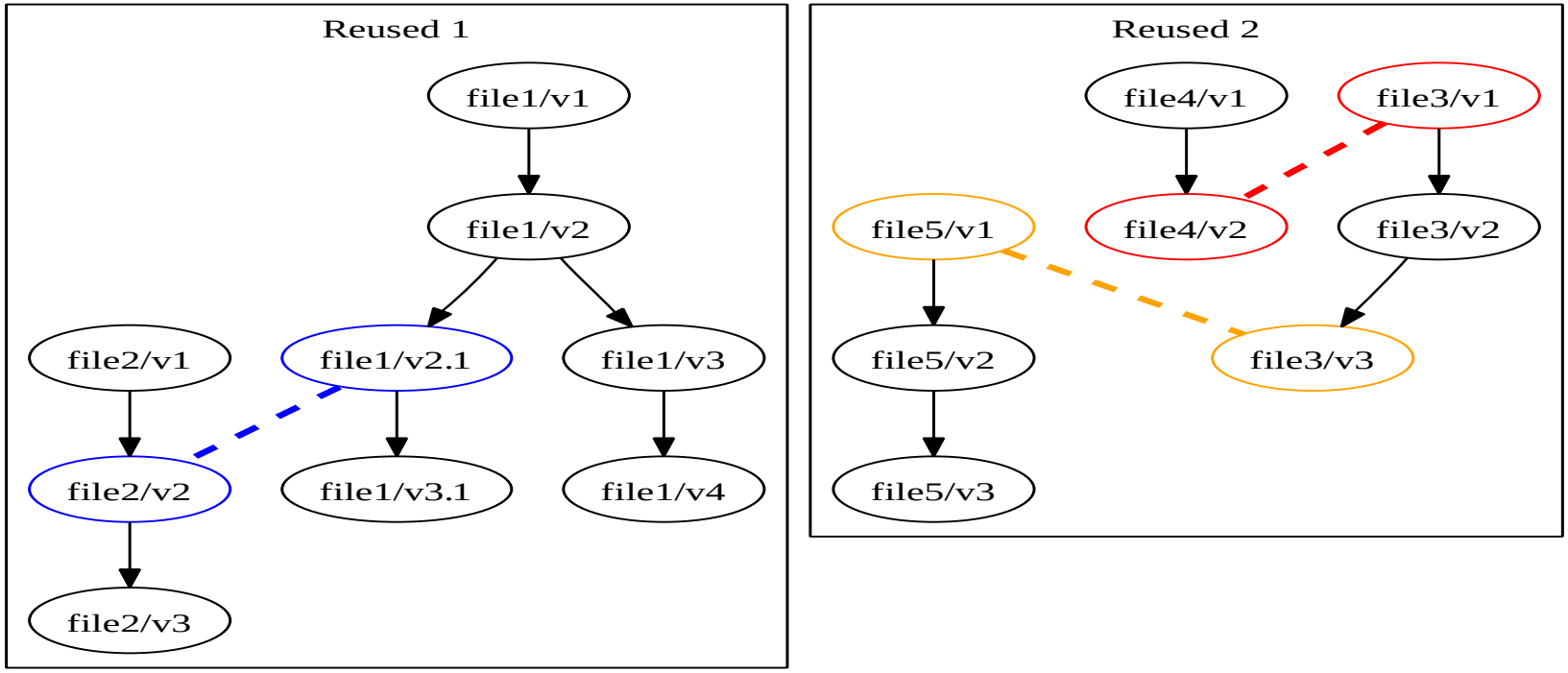
- Filename Comparison method:
  - find directory pairs with a large fraction of identical filenames
  - consider files with the same name in such directory pairs to be reused files
  - has been previously applied on Avaya codebase (>3M files) with known instances of copy
    - small errors and extensive reuse were observed
- Simple to apply — requires only the list of pathnames
- Research Questions
  - Do the results for Avaya extend to OSS projects?
  - How to validate a method with few or no known instances of reuse?

Validation: compare to methods with different properties

- Method properties
  - Data&precision: filename-based (simple) vs. content-based (link individual versions)
  - Error: low false positives vs. low false negatives
  - Technology: language specific vs. language independent
  - Domain: utilize lessons from text retrieval
- Perform expert-based validation on a smaller subsample
  - Select a manageable sample size for manual validation
  - Select candidate pairs that have the highest likelihood of errors for manual validation

Identical Content method

Reuse: the closure of files sharing at least one identical version



Construct an associative array using content from all versions of all files

- key — the content C
- value — the list of filenames and their versions that have content C

Remaining Methods: AST, Trigram, and Vector-Space

- Find the minimal distance between all versions of file A and file B using one of the following methods
  - Abstract Syntax Tree (AST): approximate AST by extracting control flow keywords and block delimiters; compare the resulting strings using a string similarity comparison
  - Nilsimsa (trigram): hash the trigrams that are accumulated from the file content into 64 digit hex code; then count the number of bytes that differ between two codes
  - Vector-Space: build term-by-document matrices and compute the similarity (cosines) between two version contents
- If the minimal distance is less than a cutoff $\Longrightarrow$ files have been reused

Characteristics of various methods

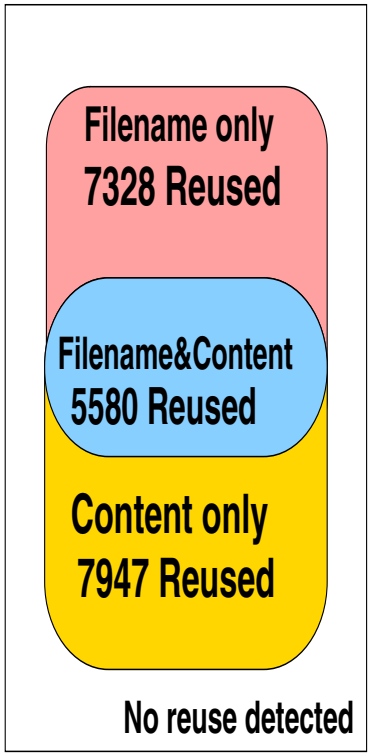
Filename	Fast: $O(D \times F(D))$. Only file path needed. Cannot match versions. Would miss cases where individual files were copied/renamed (many false negatives), but few false positives.
Identical Content	Simple and fast: $O(V \log(V))$, but needs large storage. Would miss reuse where copy involved a slightest edit (many false negatives), but almost no false positives.
Nilsimsa	Slow: $O(T^2)$. Common in text retrieval and insensitive to small changes but insensitive to source code order. May suffer from many false positives.
Vector-Space	Need to clean the content (ex: comments). Common in text retrieval, but insensitive to source code order. May suffer from many false positives.
Abstract Syntax Tree	Slow: $O(T^2)$. Language specific. Can detect control flow reuse.

FreeBSD statistics

- 492583 versions of 57128 files
- 360877 unique contents (17% fewer than versions)
- Total size of all unique contents is 8.16 G
- Average size of unique content is 16.6 ± 45.4Kb
- Average 1.365 ± 1.3 distinct filenames per unique content

FreeBSD validation

- For Nilsimsa, Vector-Space and AST methods
  1. Apply 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 sets detected by a single method to select 20 ($\frac{1}{20} = 0.05$) files.
  4. Two experts verify identified examples of reuse.



Results of validation

- Remove 9569 non-C files leaving 47559 C-language files

Filename Comparison:	12908	Reused Files	27%
Identical Content:	13077	Reused Files	27%
Filename only:	7328	Reused Files	15%
Filename & Content:	5580	Reused Files	12%
Only Content:	7497	Reused Files	16%

- Total number Reused Files is 43 % ((7328+5580+7947) / 47559 = 43%)
- Nilsimsa, Vector-Space and AST methods on Filename & Content cases

AST	3027	Reused Files	15%
Nilsimsa	3143	Reused Files	15%
Vector-Space	1120	Reused Files	5%

Manual inspection

- Many Reused Files are missed by Identical Content method due to minor content changes (including comments)
- Inspection by two experts:
  - Sample 20 reuse pairs from each method (AST, Vector, trigram)

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

  - AST: Non-Reused Files were not C-language code
  - Nilsimsa: appears to match primarily on the copyright notice
  - Vector-Space: may be not particularly suited for copy detection

Summary

- It appears to be possible to estimate and validate reuse and link multiple change histories:
  - without known instances to train the method,
  - by relying on different properties of each method.
- Filename Comparison:
  - easy-to-apply, similar behavior on proprietary and OSS code
    - detects 60% of reused files,
    - with a 4% false-positive rate.
- Some computational challenges remain for large-scale data (e.g., OSS, with 60M files).

Acknowledgments

Ahmed Hassan (AST approximation code)

References

[1] Hung-Fu Chang and Audris Mockus. Constructing universal version history. In *ICSE'06 Workshop on Mining Software Repositories*, pages 76–79, Shanghai, China, May 22-23 2006.