

Detecting Source-Code Plagiarism

Mike Joy

Department of Computer Science, University of Warwick

<http://www.dcs.warwick.ac.uk/research/edtech/>

What is Source-code Plagiarism?

“Plagiarism occurs when programming assignments are copied and transformed with very little effort from the students”

(Faidhi and Robinson, 1987)

“unacknowledged copying of documents and programs”

(Joy and Luck, 1999)

Survey on Source-Code Plagiarism

We carried out a survey in order to:

- gather the perceptions of academics on what constitutes source-code plagiarism, and
- create a structured description of what constitutes source-code plagiarism from a UK academic perspective

Short paper:

www.dcs.warwick.ac.uk/research/edtech/publications/CosmaJoy06.pdf

Full report: www.dcs.warwick.ac.uk/reports/422.html

Data Source

- On-line questionnaire distributed to 120 academics (list supplied by HEA-ICS).
- Survey:
 - Questions were in the form of small scenarios
 - Mostly multiple-choice responses
 - Comments box below each question
 - Anonymous – option for providing details
- Received 59 responses, from academics across more than 34 different institutions
- Responses were analysed and collated to create a universally acceptable source-code plagiarism description.

Source-Code Plagiarism Issues

Our description considers various issues on source-code plagiarism including:

- Source-code reuse
- Source-code self-plagiarism
- Copying without adaptation
- Copying with adaptation: minimal, moderate, extreme
- Converting a source-code to another programming language
- Using code-generator software
- Collusion
- Methods of obtaining source-code written by other authors
- False and “pretend” references

Survey Findings: Reuse

Two **grey areas** regarding source-code reuse and self-plagiarism:

- Reuse of code in O-O environments is often encouraged
- Source code resubmission may be acceptable

Historical Notes

- First known plagiarism detection system was an attribute counting program developed by Ottenstein (1976)
- More recent systems compare the structure of source-code programs
- Structure-based systems include: YAP3, MOSS, JPlag, Plague, and Sherlock.

Detection Tools

Structure-based systems:

1. Each program is converted into token strings (or something similar)
2. Token streams are compared for determining similar source-code fragments

3 Tools: JPlag, MOSS, and Sherlock

JPlag

- Developed by Guido Malpohl in 1996
- JPlag currently supports Java, C#, C, C++, Scheme, and natural language text
- Use of JPlag is free, but user must create an account
- JPlag can be used to compare student assignments but does not compare with code on the Internet
- JPlag home page: www.ipd.uni-karlsruhe.de/jplag

JPlag File Processing

JPlag Web Start client v2.0 BETA - Build 293

Main menu Help

Title	Submitted	Language	Number of programs	Parser errors
Untitled1	02.05.2006 17:50:47	java12	168	22
cs123	19.05.2006 11:19:08	java12	202	26

JPlag progress

- Packing files
- Sending files
- Waiting in queue
- Parsing files
- Comparing files
- Downloading results

Submission: cs123
Operation was successful.
Result location:
C:\Documents and Settings\Georgina\JPlag_Results\cs123\index.html

Close

Hide progress

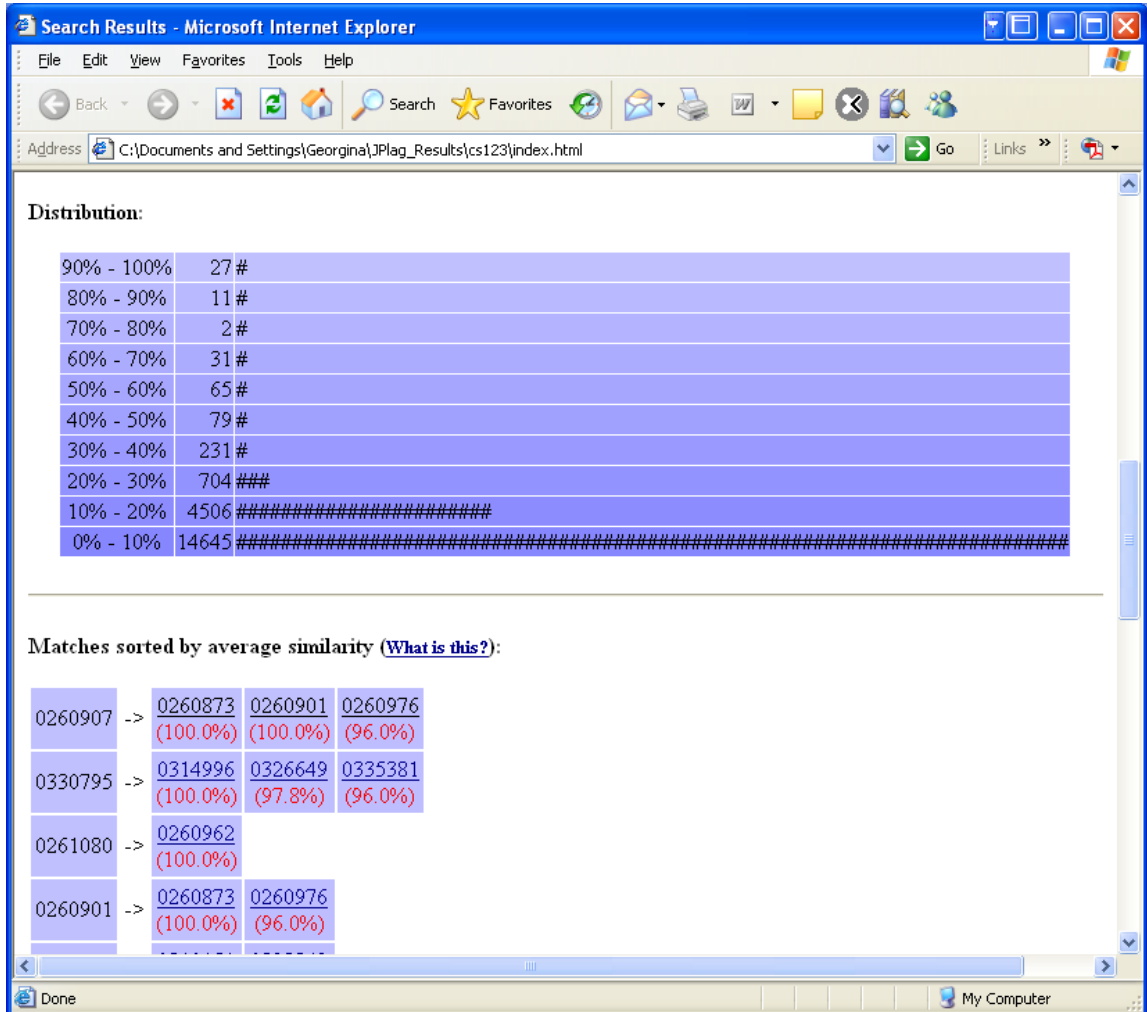
New submission Delete result

View result Change options

View parser log Rename result

JPlag - Results

- Results in HTML Format
- Histogram of similarity values found for all pairs of programs
- Similar pairs and their similarity values displayed
- Select file pairs to view



JPlag - Matches

- Similar lines matched with the same colour
- Code fragment similarity values based on similar tokens found

Matches for 0260976 & 0260907

96.0%

[INDEX](#) - [HELP](#)

	0260976 (96.0%)	0260907 (96.0%)	Tokens
	1079088194000/Board.java(1-22)	1079082729000/Board.java(1-26)	19
	1079088194000/Board.java(24-67)	1079082729000/Board.java(28-77)	36
	1079088194000/Board.java(70-74)	1079082729000/Board.java(78-84)	11
	1079088194000/Board.java(75-149)	1079082729000/Board.java(87-173)	112
	1079088194000/Board.java(149-167)	1079082729000/Board.java(176-195)	23

```
package student;
import wrabble.*;
import java.util.StringTokenizer;

public class Board implements IBoard
{
    protected Multiplier [][]multipliers;
    protected Tile [][]tiles;
    protected int size;

    public Board()
    {
        size = 15;
        multipliers = new Multiplier [size][size];
        tiles = new Tile [size][size];
    }

    public void clearBoard()
    {
        for(int i = 0;i < 15;i++)
        {
            for(int j = 0;j < 15; j++)
            {
                tiles[i][j] = null;
            }
        }
    }

    public Tile getFile(int x,int y)
    {
    }
}
```

```
package student;
import wrabble.*;
import java.util.StringTokenizer;

public class Board implements IBoard
{
    protected Multiplier[][] multi;
    protected Tile[][] tile;
    protected int size;

    public Board()
    {
        size = 15;
        multi = new Multiplier [size][size];
        tile = new Tile [size][size];
    }

    public void clearBoard()
    {
        for(int i = 0;i < getBoardSize();i++)
        {
            for(int j = 0;j < getBoardSize(); j++)
            {
                tile[i][j] = null;
            }
        }
    }
}
```

MOSS

Developed by Alex Aiken in 1994

MOSS (for a Measure Of Software Similarity)
determines the similarity of C, C++, Java, Pascal,
Ada, ML, Lisp, or Scheme programs.

MOSS is free, but you must create an account

MOSS home page:

www.cs.berkeley.edu/~aiken/moss.html

Using MOSS

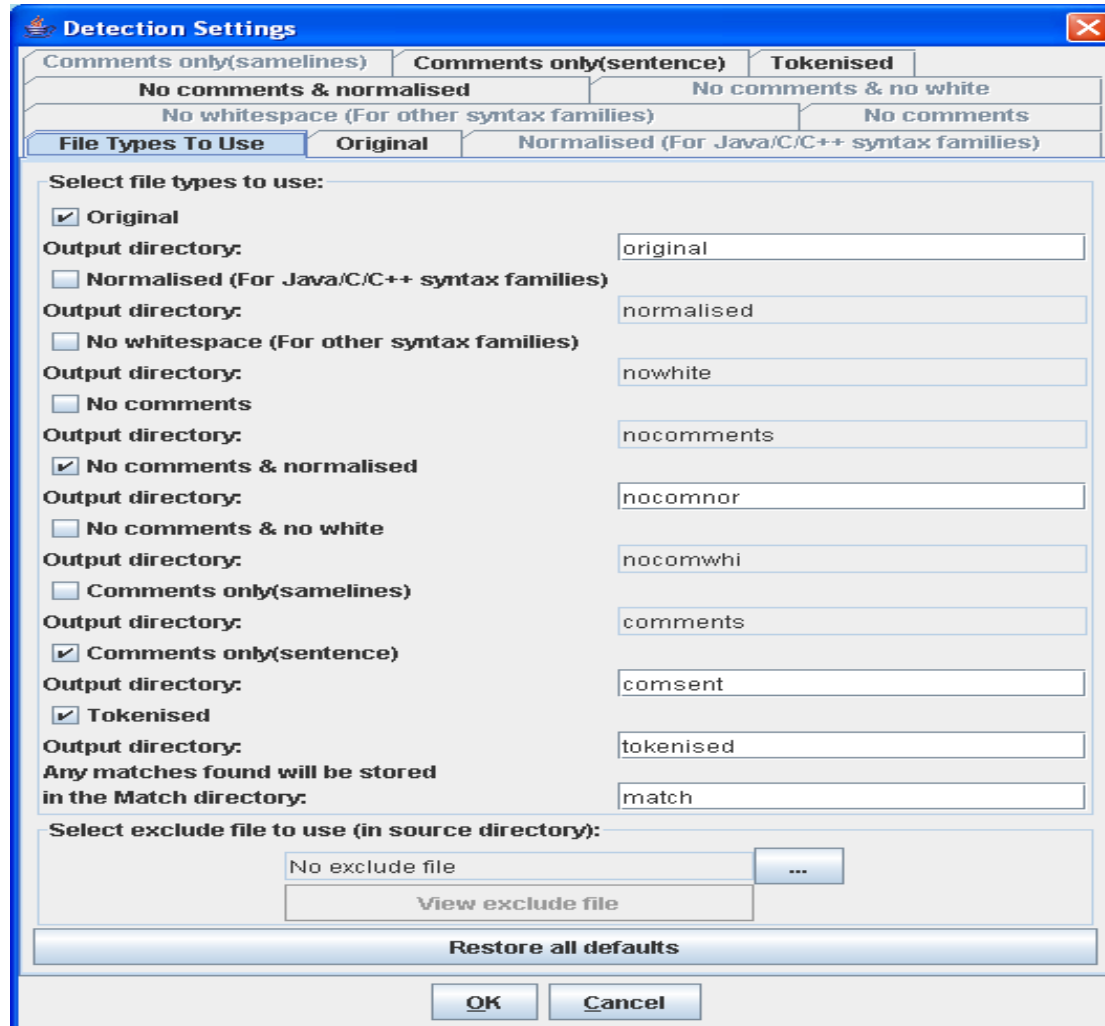
- Moss is being provided as an Internet service
- User must download MOSS Perl script for submitting files to the MOSS server
- The script uses a direct network connection
- The MOSS server produces HTML pages listing pairs of programs with similar code
- MOSS highlights similar code-fragments within programs that appear the same
- Data Protection? – US service
- Maintenance?

Sherlock

- Developed at the University of Warwick Department of Computer Science
- Sherlock was fully integrated with the BOSS online submission software in 2002 and Open-Sourced
- Sherlock detects plagiarism on source-code and natural language assignments
- BOSS home page: www.boss.org.uk

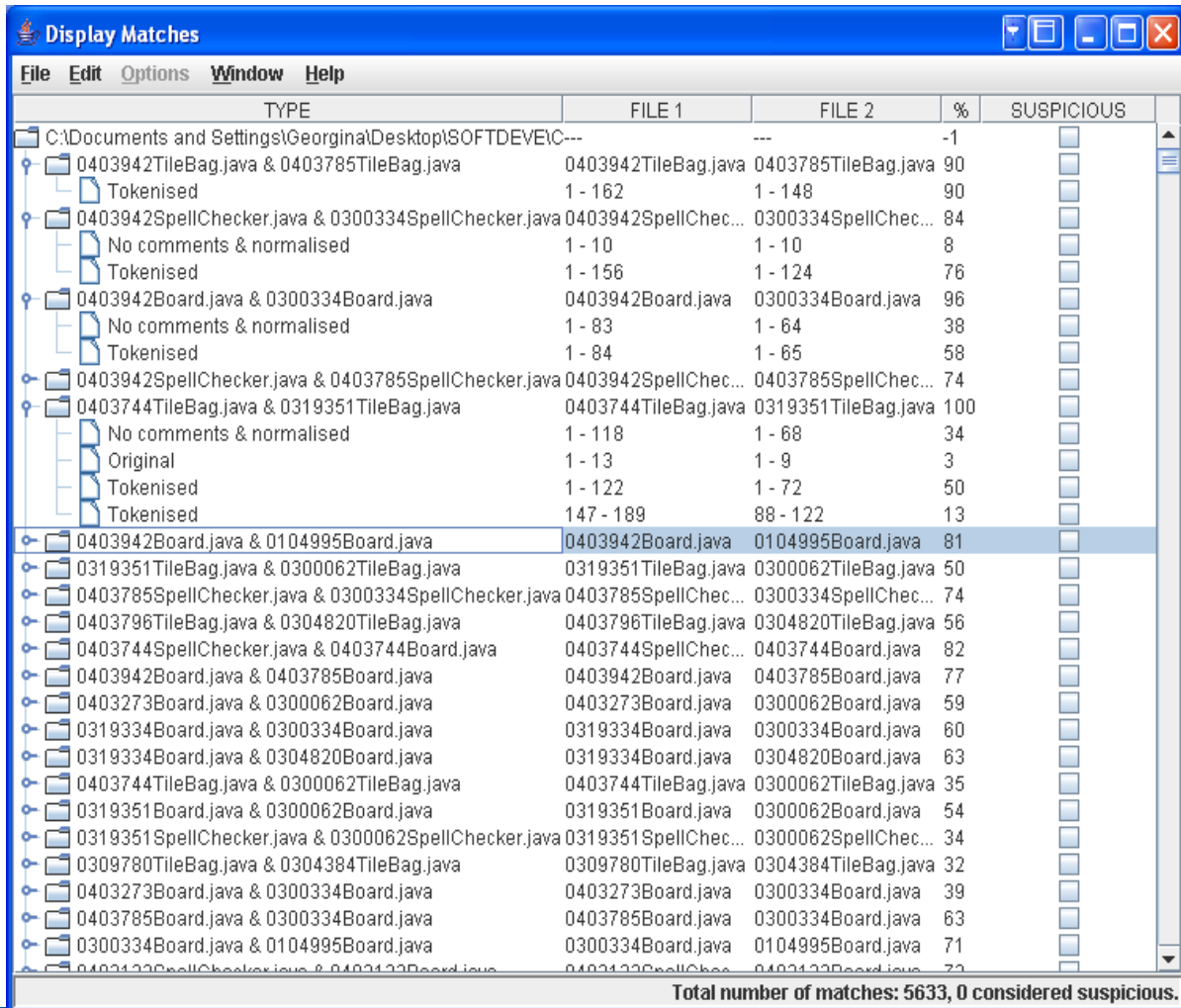
Sherlock - Preprocessing

Whitespace
Comments
Normalisation
Tokenisation



Sherlock - Results

- Results displayed
- Similarity values of suspicious files
- Similarity values depend on the length of similar lines found as a percentage of the whole file size
- Select suspicious matches to examine
- Mark suspicious files



The screenshot shows a window titled "Display Matches" with a menu bar (File, Edit, Options, Window, Help) and a table of file comparison results. The table has columns for TYPE, FILE 1, FILE 2, %, and SUSPICIOUS. The SUSPICIOUS column contains checkboxes. The table lists various file pairs and their similarity percentages. The entry "0403942Board.java & 0104995Board.java" is highlighted, showing a similarity of 81%.

TYPE	FILE 1	FILE 2	%	SUSPICIOUS
C:\Documents and Settings\Georgina\Desktop\SOFTDEV\IC---	---	---	-1	<input type="checkbox"/>
0403942TileBag.java & 0403785TileBag.java	0403942TileBag.java	0403785TileBag.java	90	<input type="checkbox"/>
Tokenised	1 - 162	1 - 148	90	<input type="checkbox"/>
0403942SpellChecker.java & 0300334SpellChecker.java	0403942SpellChec...	0300334SpellChec...	84	<input type="checkbox"/>
No comments & normalised	1 - 10	1 - 10	8	<input type="checkbox"/>
Tokenised	1 - 156	1 - 124	76	<input type="checkbox"/>
0403942Board.java & 0300334Board.java	0403942Board.java	0300334Board.java	96	<input type="checkbox"/>
No comments & normalised	1 - 83	1 - 64	38	<input type="checkbox"/>
Tokenised	1 - 84	1 - 65	58	<input type="checkbox"/>
0403942SpellChecker.java & 0403785SpellChecker.java	0403942SpellChec...	0403785SpellChec...	74	<input type="checkbox"/>
0403744TileBag.java & 0319351TileBag.java	0403744TileBag.java	0319351TileBag.java	100	<input type="checkbox"/>
No comments & normalised	1 - 118	1 - 68	34	<input type="checkbox"/>
Original	1 - 13	1 - 9	3	<input type="checkbox"/>
Tokenised	1 - 122	1 - 72	50	<input type="checkbox"/>
Tokenised	147 - 189	88 - 122	13	<input type="checkbox"/>
0403942Board.java & 0104995Board.java	0403942Board.java	0104995Board.java	81	<input type="checkbox"/>
0319351TileBag.java & 0300062TileBag.java	0319351TileBag.java	0300062TileBag.java	50	<input type="checkbox"/>
0403785SpellChecker.java & 0300334SpellChecker.java	0403785SpellChec...	0300334SpellChec...	74	<input type="checkbox"/>
0403796TileBag.java & 0304820TileBag.java	0403796TileBag.java	0304820TileBag.java	56	<input type="checkbox"/>
0403744SpellChecker.java & 0403744Board.java	0403744SpellChec...	0403744Board.java	82	<input type="checkbox"/>
0403942Board.java & 0403785Board.java	0403942Board.java	0403785Board.java	77	<input type="checkbox"/>
0403273Board.java & 0300062Board.java	0403273Board.java	0300062Board.java	59	<input type="checkbox"/>
0319334Board.java & 0300334Board.java	0319334Board.java	0300334Board.java	60	<input type="checkbox"/>
0319334Board.java & 0304820Board.java	0319334Board.java	0304820Board.java	63	<input type="checkbox"/>
0403744TileBag.java & 0300062TileBag.java	0403744TileBag.java	0300062TileBag.java	35	<input type="checkbox"/>
0319351Board.java & 0300062Board.java	0319351Board.java	0300062Board.java	54	<input type="checkbox"/>
0319351SpellChecker.java & 0300062SpellChecker.java	0319351 SpellChec...	0300062SpellChec...	34	<input type="checkbox"/>
0309780TileBag.java & 0304384TileBag.java	0309780TileBag.java	0304384TileBag.java	32	<input type="checkbox"/>
0403273Board.java & 0300334Board.java	0403273Board.java	0300334Board.java	39	<input type="checkbox"/>
0403785Board.java & 0300334Board.java	0403785Board.java	0300334Board.java	63	<input type="checkbox"/>
0300334Board.java & 0104995Board.java	0300334Board.java	0104995Board.java	71	<input type="checkbox"/>
0403122SpellChecker.java & 0403122Board.java	0403122SpellChec...	0403122Board.java	72	<input type="checkbox"/>

Total number of matches: 5633, 0 considered suspicious.

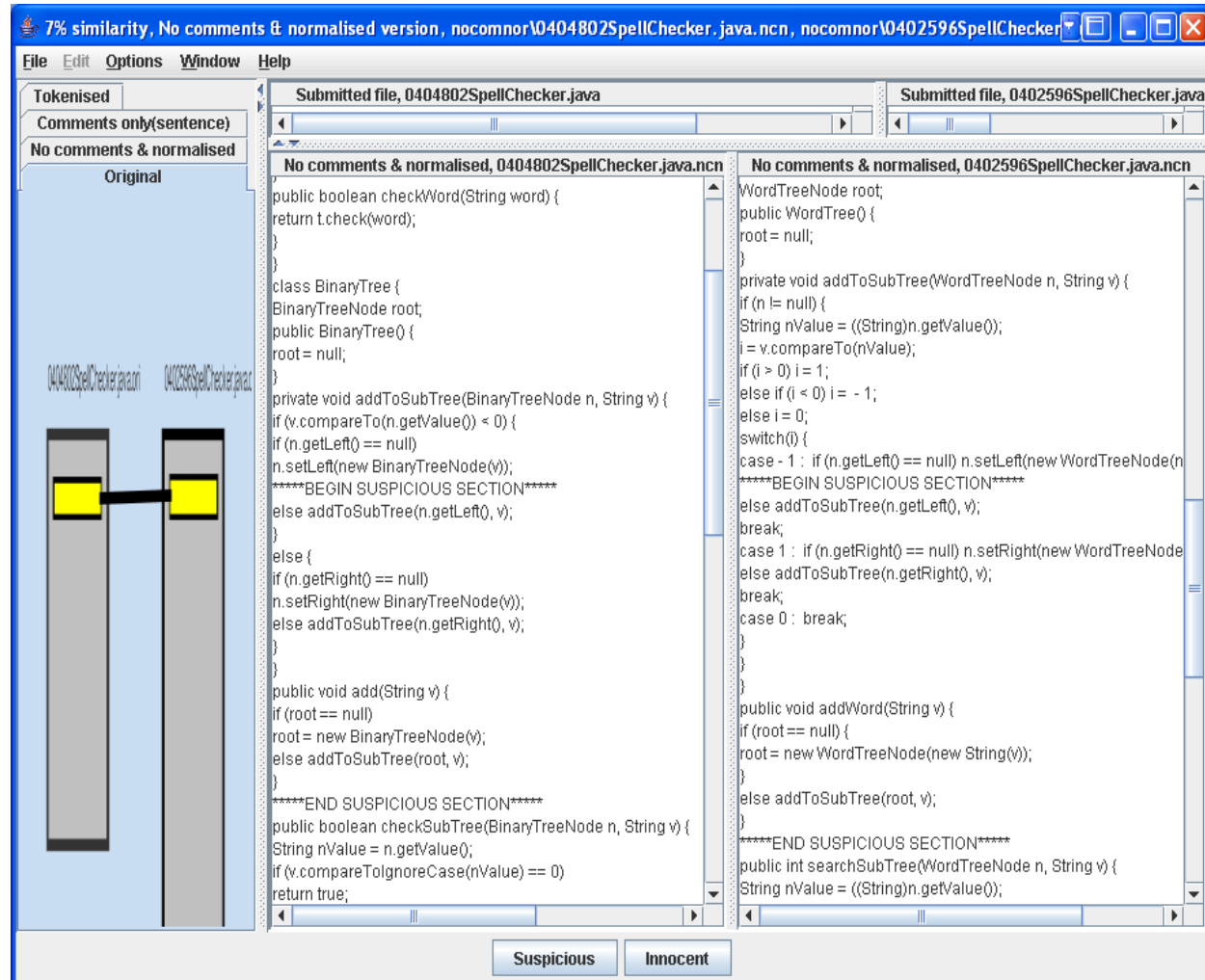
Sherlock - Matches

Suspected sections marked with

****begin suspicious section****

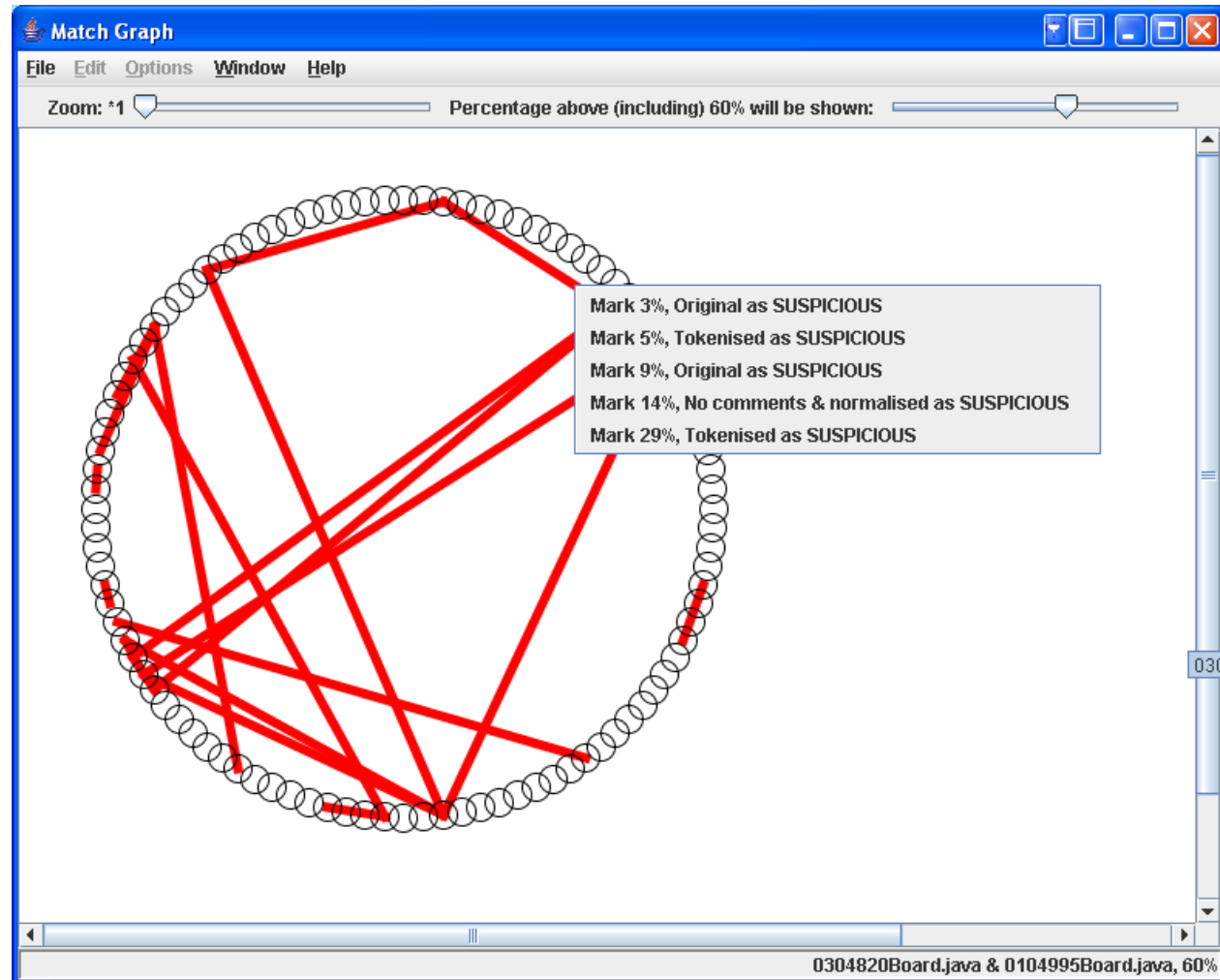
and

****end suspicious section****



Sherlock – Document Set

- User can view graph
- Each node represents one submission
- An edge means two submissions
- Options to select threshold
- Click on lines to view or to mark suspicious matches



Tool Efficiency

- MOSS, JPlag and Sherlock are effective
- Results returned are not identical
- User interface issues may be important

Conclusion

- Important to communicate clearly to students what constitutes source-code plagiarism within a broad, and where appropriate, an assignment specific context (for example, state explicitly whether source-code reuse is encouraged)
- Several free plagiarism detection tools are available
- Important to **use** the tools