Catch Me If You Can: Detecting Plagiarism Using Code Similarity Detection Tools

Sang Jun Chun, Claire Luo

Background

- Motivation: Find the most effective code similarity detection tools for catching plagiarism in school projects.
- Tools (open source) we used:
  - **Plaggie:** source code similarity detector
  - **DeepSim & GEMINI:** binary code similarity detector
  - **SIM:** text difference detector

Research Questions

RQ1: How effectively can code similarity tools detect **different types of code plagiarism**?

RQ2: How do different types of code similarity tools react to **original works** addressing the same problem?

RQ3: Of the approaches examined, **which** is particularly effective for detecting academic plagiarism in code?

Methodology (1/2)

- 37 Java homework submissions for COMS W4156 (Fall 2020)
  - Only a subset of the entire pool available as public repositories on GitHub
- How to label pairs?

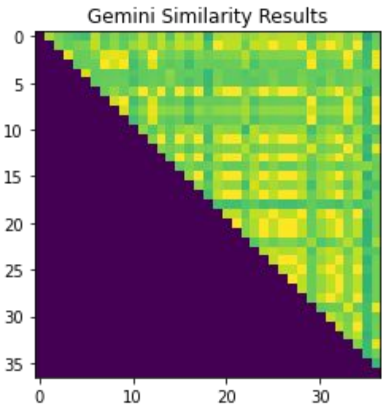
GEMINI: Cosine similarity on embedding	[0,1]
DeepSim: Confidence on classification on semantic embedding	[0,1]
Plaggie: Generate token-level similarity %, and default threshold set at 50%	%
SIM: Text diff similarity %, threshold set by the artificial pairs (59.8%).	%

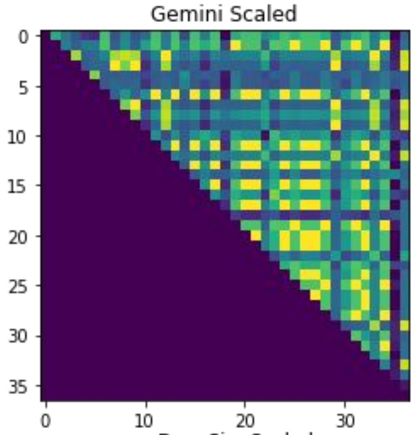
- True label
  - Manual detection

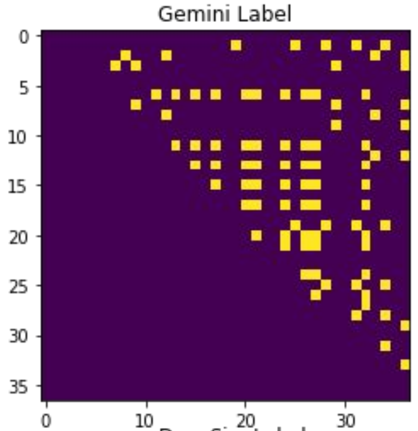
Methodology (2/2)

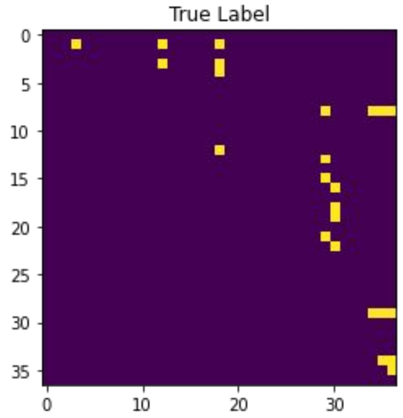
- Artificial plagiarism
  - Created plagiarized code using a team member’s original submission
  - Three categories:
    - Variables change (name, type)
    - Line change (insert or remove verbose lines)
    - Structure change (rearrange the ordering of classes/methods)

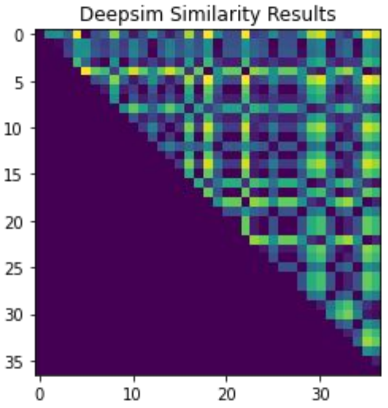
Results - DeepSim & GEMINI

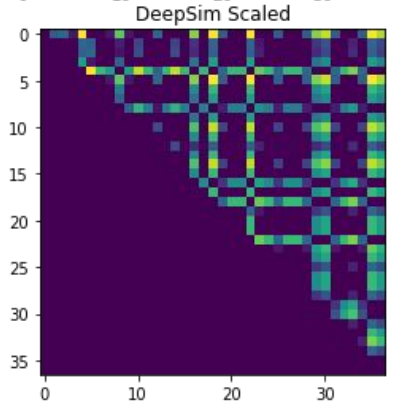


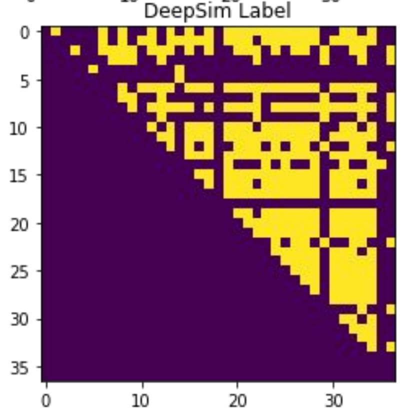




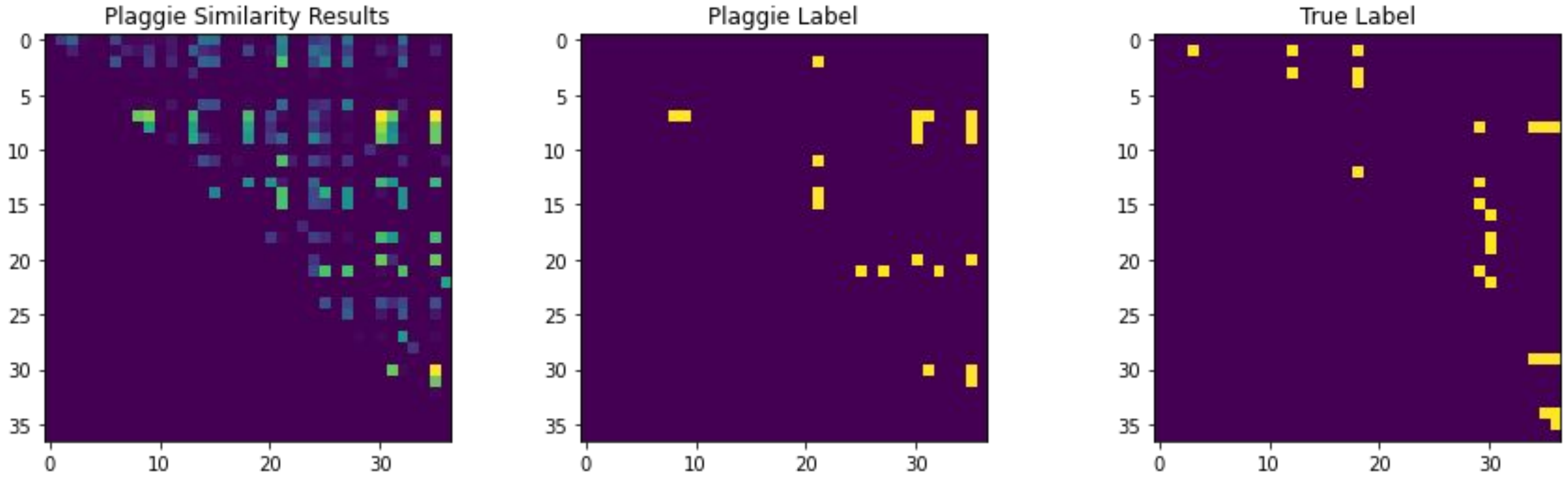




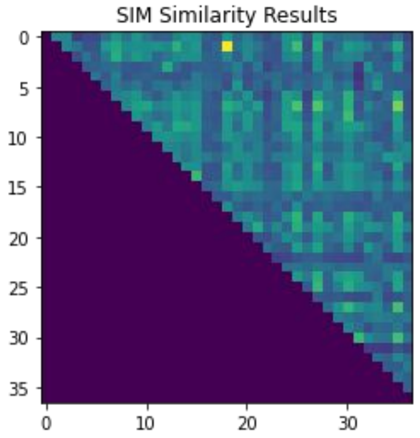


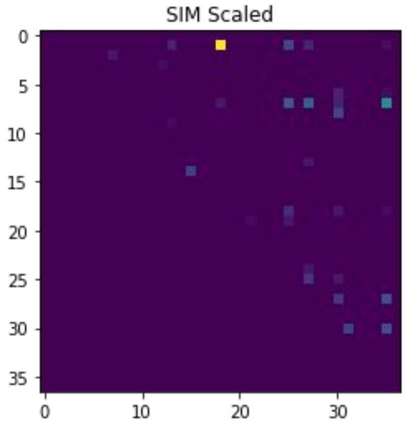


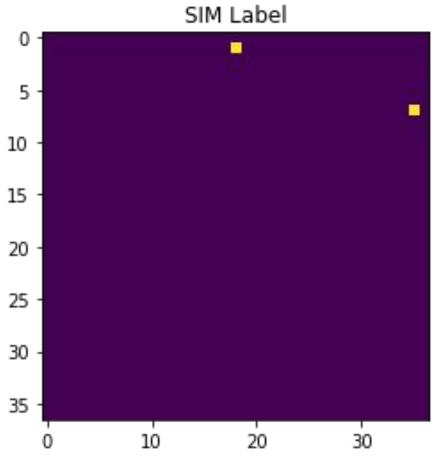
Results - Plaggie

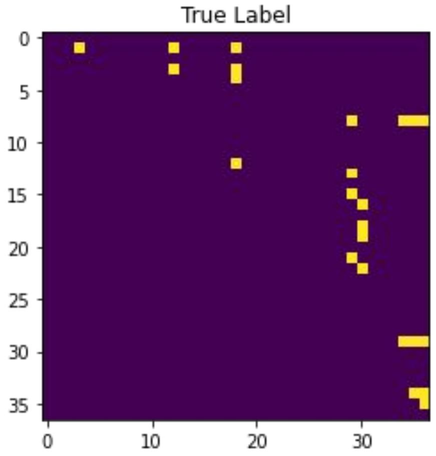


Results - SIM


SIM Similarity Results
0
5
10
15
20
25
30
35
0
10
20
30


SIM Scaled
0
5
10
15
20
25
30
35
0
10
20
30


SIM Label
0
5
10
15
20
25
30
35
0
10
20
30

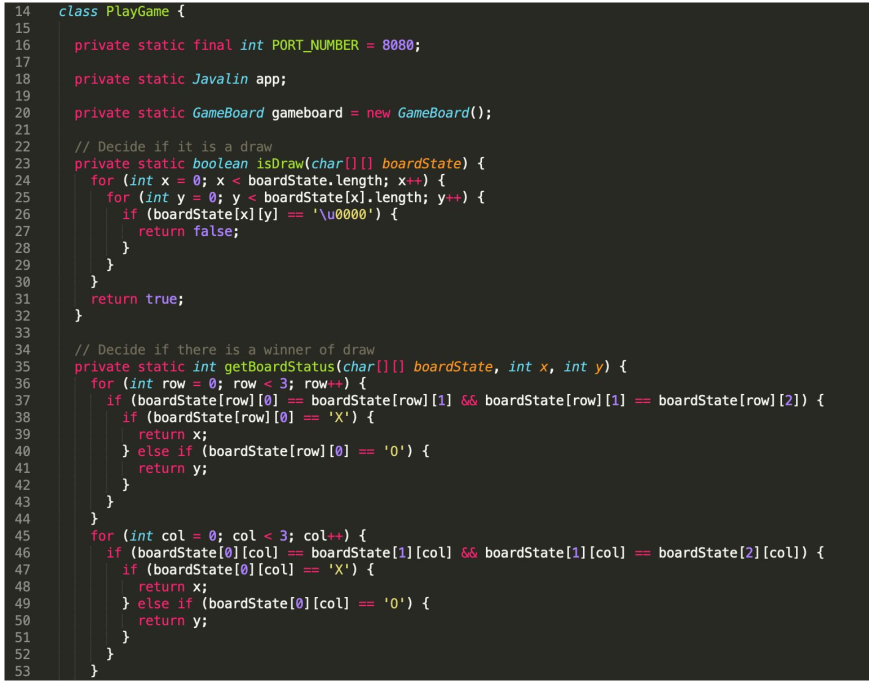

True Label
0
5
10
15
20
25
30
35
0
10
20
30

Aggregate Results

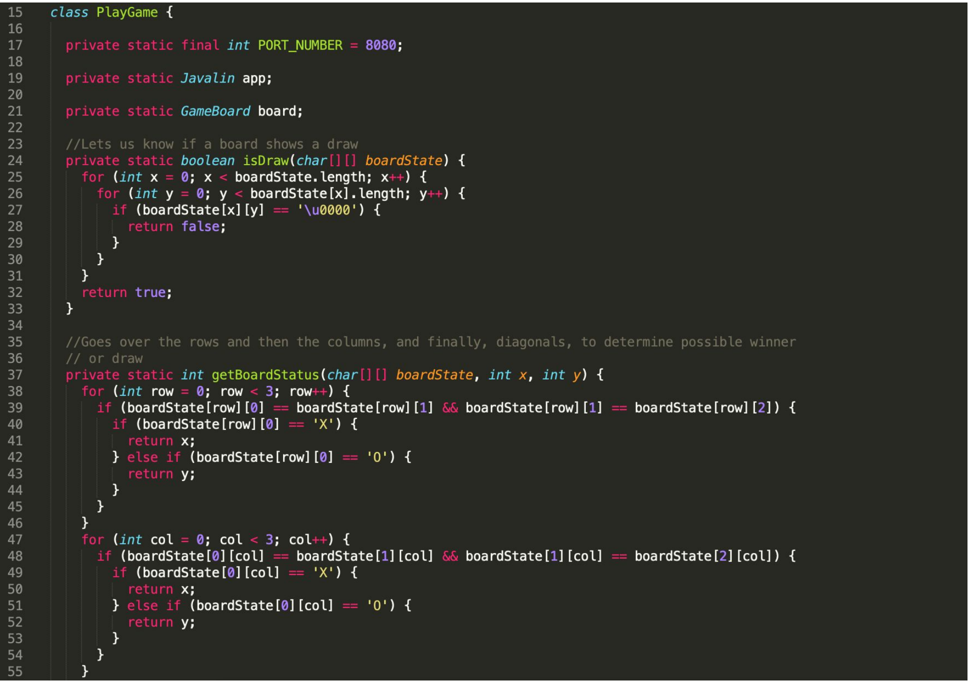
Pairs	**DeepSim/GEMINI**	**Plaggie**	**SIM**	**Plagiarized?***
(2, 19)	0.492; 0.998	47.1%	78%	Likely
(9, 36)	0.568; 0.981	60.9%	63%	Likely
(31, 36)	0.268;0.6924	56.2%	55%	Unlikely
(2,29)	0.360; 0.9056	32.6%	21%	Unlikely
(6,37)	0.870; 0.988	40.8%	22%	Unlikely
(8,31)	0.5947; 0.7517	56.2%	50%	Unlikely

*Manually inspected.

An Example of Plagiarism Caught (SIM)

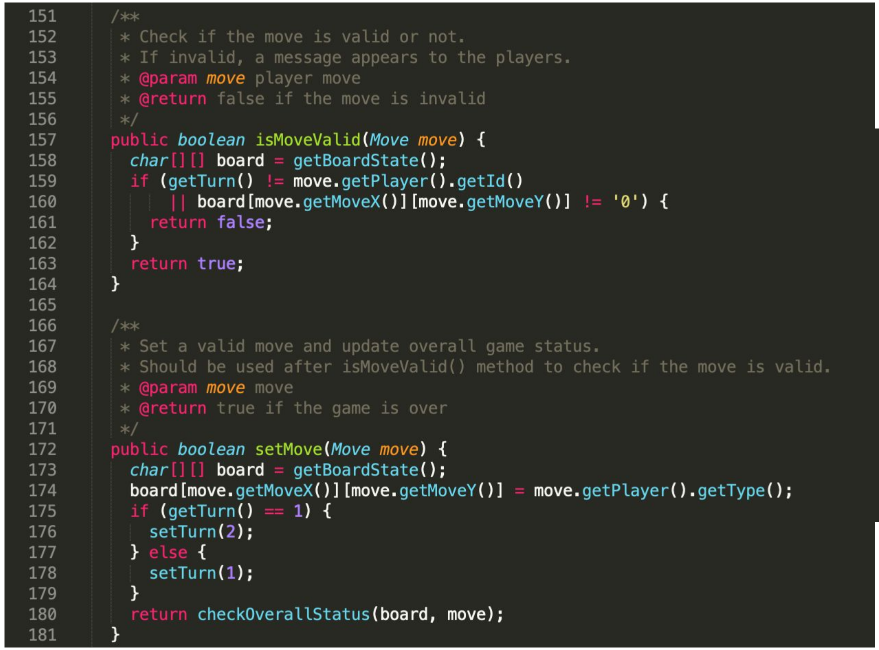


```java
class PlayGame {

  private static final int PORT_NUMBER = 8080;

  private static Javalin app;

  private static GameBoard gameboard = new GameBoard();

  // Decide if it is a draw
  private static boolean isDraw(char[][] boardState) {
    for (int x = 0; x < boardState.length; x++) {
      for (int y = 0; y < boardState[x].length; y++) {
        if (boardState[x][y] == '\u0000') {
          return false;
        }
      }
    }
    return true;
  }

  // Decide if there is a winner of draw
  private static int getBoardStatus(char[][] boardState, int x, int y) {
    for (int row = 0; row < 3; row++) {
      if (boardState[row][0] == boardState[row][1] && boardState[row][1] == boardState[row][2]) {
        if (boardState[row][0] == 'X') {
          return x;
        } else if (boardState[row][0] == 'O') {
          return y;
        }
      }
    }
    for (int col = 0; col < 3; col++) {
      if (boardState[0][col] == boardState[1][col] && boardState[1][col] == boardState[2][col]) {
        if (boardState[0][col] == 'X') {
          return x;
        } else if (boardState[0][col] == 'O') {
          return y;
        }
      }
    }
```

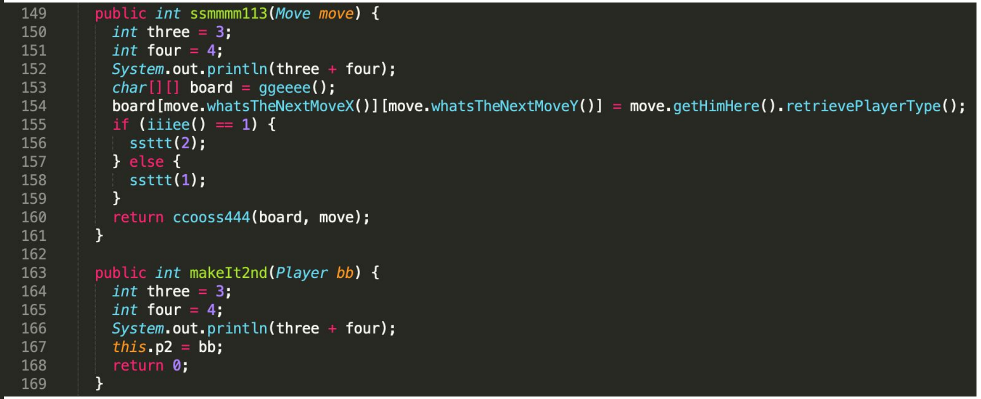


```java
class PlayGame {

  private static final int PORT_NUMBER = 8080;

  private static Javalin app;

  private static GameBoard board;

  //Lets us know if a board shows a draw
  private static boolean isDraw(char[][] boardState) {
    for (int x = 0; x < boardState.length; x++) {
      for (int y = 0; y < boardState[x].length; y++) {
        if (boardState[x][y] == '\u0000') {
          return false;
        }
      }
    }
    return true;
  }

  //Goes over the rows and then the columns, and finally, diagonals, to determine possible winner
  // or draw
  private static int getBoardStatus(char[][] boardState, int x, int y) {
    for (int row = 0; row < 3; row++) {
      if (boardState[row][0] == boardState[row][1] && boardState[row][1] == boardState[row][2]) {
        if (boardState[row][0] == 'X') {
          return x;
        } else if (boardState[row][0] == 'O') {
          return y;
        }
      }
    }
    for (int col = 0; col < 3; col++) {
      if (boardState[0][col] == boardState[1][col] && boardState[1][col] == boardState[2][col]) {
        if (boardState[0][col] == 'X') {
          return x;
        } else if (boardState[0][col] == 'O') {
          return y;
        }
      }
    }
```

An Example of Plagiarism Caught (DeepSim/Gemini)



```java
/**
 * Check if the move is valid or not.
 * If invalid, a message appears to the players.
 * @param move player move
 * @return false if the move is invalid
 */
public boolean isMoveValid(Move move) {
  char[][] board = getBoardState();
  if (getTurn() != move.getPlayer().getId()
      || board[move.getMoveX()][move.getMoveY()] != '0') {
    return false;
  }
  return true;
}

/**
 * Set a valid move and update overall game status.
 * Should be used after isMoveValid() method to check if the move is valid.
 * @param move move
 * @return true if the game is over
 */
public boolean setMove(Move move) {
  char[][] board = getBoardState();
  board[move.getMoveX()][move.getMoveY()] = move.getPlayer().getType();
  if (getTurn() == 1) {
    setTurn(2);
  } else {
    setTurn(1);
  }
  return checkOverallStatus(board, move);
}
```



```java
public int ssmmmm113(Move move) {
  int three = 3;
  int four = 4;
  System.out.println(three + four);
  char[][] board = ggeeee();
  board[move.whatsTheNextMoveX()][move.whatsTheNextMoveY()] = move.getHimHere().retrievePlayerType();
  if (iiiee() == 1) {
    ssttt(2);
  } else {
    ssttt(1);
  }
  return ccooss444(board, move);
}

public int makeIt2nd(Player bb) {
  int three = 3;
  int four = 4;
  System.out.println(three + four);
  this.p2 = bb;
  return 0;
}
```

Analysis (1/2)

RQ1: How effectively can code similarity tools detect different types of code plagiarism?

- Compare similarity scores for artificially plagiarized code

	Lines removed	Variables changed	Structures changed	All changed
GEMINI	0.877	0.972	0.999	0.8948
Plaggie	50.2%	77.9%	68.8%	46.1%
SIM	49%	79%	92%	41%

Analysis (2/2)

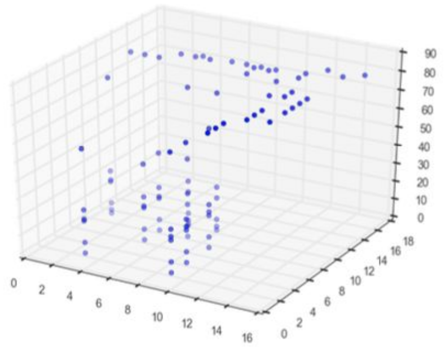
RQ2: How do different types of code similarity tools react to original works addressing the same problem?

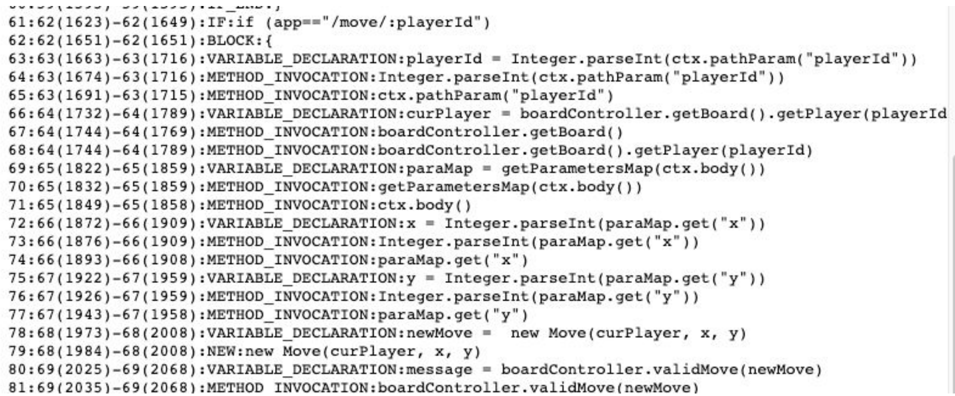
Tool	Similarity score
DeepSim/GEMINI	0.73 / 0.656
Plaggie	30.5%
SIM	47%

RQ3: Of the approaches examined, which is particularly effective for detecting academic plagiarism in code?

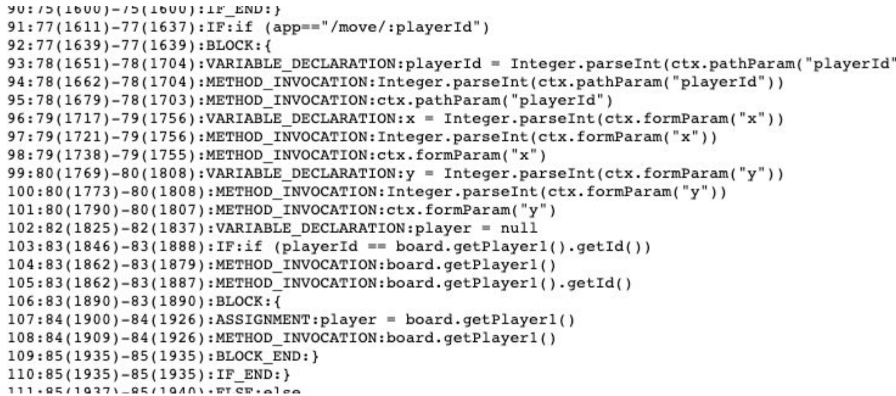
Takeaways

- Binary code similarity detectors performed well, but produced many false positives
  - Possible reasons: skeleton code, lack of data points, modified code
- Plaggie (source code similarity detector) is only good at catching obvious copies
- Adding or removing redundant lines from the original code hinders the tools the most





```
60:59(1595)-59(1595):IF_END:}
61:62(1623)-62(1649):IF:if (app=="/move/:playerId")
62:62(1651)-62(1651):BLOCK:{
63:63(1663)-63(1716):VARIABLE_DECLARATION:playerId = Integer.parseInt(ctx.pathParam("playerId"))
64:63(1674)-63(1716):METHOD_INVOCATION:Integer.parseInt(ctx.pathParam("playerId"))
65:63(1691)-63(1715):METHOD_INVOCATION:ctx.pathParam("playerId")
66:64(1732)-64(1789):VARIABLE_DECLARATION:curPlayer = boardController.getBoard().getPlayer(playerId
67:64(1744)-64(1769):METHOD_INVOCATION:boardController.getBoard()
68:64(1744)-64(1789):METHOD_INVOCATION:boardController.getBoard().getPlayer(playerId)
69:65(1822)-65(1859):VARIABLE_DECLARATION:paraMap = getParametersMap(ctx.body())
70:65(1832)-65(1859):METHOD_INVOCATION:getParametersMap(ctx.body())
71:65(1849)-65(1858):METHOD_INVOCATION:ctx.body()
72:66(1872)-66(1909):VARIABLE_DECLARATION:x = Integer.parseInt(paraMap.get("x"))
73:66(1876)-66(1909):METHOD_INVOCATION:Integer.parseInt(paraMap.get("x"))
74:66(1893)-66(1908):METHOD_INVOCATION:paraMap.get("x")
75:67(1922)-67(1959):VARIABLE_DECLARATION:y = Integer.parseInt(paraMap.get("y"))
76:67(1926)-67(1959):METHOD_INVOCATION:Integer.parseInt(paraMap.get("y"))
77:67(1943)-67(1958):METHOD_INVOCATION:paraMap.get("y")
78:68(1973)-68(2008):VARIABLE_DECLARATION:newMove =  new Move(curPlayer, x, y)
79:68(1984)-68(2008):NEW:new Move(curPlayer, x, y)
80:69(2025)-69(2068):VARIABLE_DECLARATION:message = boardController.validMove(newMove)
81:69(2035)-69(2068):METHOD_INVOCATION:boardController.validMove(newMove)
```



```
90:75(1600)-75(1600):IF_END:}
91:77(1611)-77(1637):IF:if (app=="/move/:playerId")
92:77(1639)-77(1639):BLOCK:{
93:78(1651)-78(1704):VARIABLE_DECLARATION:playerId = Integer.parseInt(ctx.pathParam("playerId"
94:78(1662)-78(1704):METHOD_INVOCATION:Integer.parseInt(ctx.pathParam("playerId"))
95:78(1679)-78(1703):METHOD_INVOCATION:ctx.pathParam("playerId")
96:79(1717)-79(1756):VARIABLE_DECLARATION:x = Integer.parseInt(ctx.formParam("x"))
97:79(1721)-79(1756):METHOD_INVOCATION:Integer.parseInt(ctx.formParam("x"))
98:79(1738)-79(1755):METHOD_INVOCATION:ctx.formParam("x")
99:80(1769)-80(1808):VARIABLE_DECLARATION:y = Integer.parseInt(ctx.formParam("y"))
100:80(1773)-80(1808):METHOD_INVOCATION:Integer.parseInt(ctx.formParam("y"))
101:80(1790)-80(1807):METHOD_INVOCATION:ctx.formParam("y")
102:82(1825)-82(1837):VARIABLE_DECLARATION:player = null
103:83(1846)-83(1888):IF:if (playerId == board.getPlayer1().getId())
104:83(1862)-83(1879):METHOD_INVOCATION:board.getPlayer1()
105:83(1862)-83(1887):METHOD_INVOCATION:board.getPlayer1().getId()
106:83(1890)-83(1890):BLOCK:{
107:84(1900)-84(1926):ASSIGNMENT:player = board.getPlayer1()
108:84(1909)-84(1926):METHOD_INVOCATION:board.getPlayer1()
109:85(1935)-85(1935):BLOCK_END:}
110:85(1935)-85(1935):IF_END:}
111:85(1937)-85(1940):ELSE:else
```