GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Code Similarity and Plagiarism Detection

Presenter: **Mehmed Mustafa**

Advisor: **Ella Albrecht**

Seminar: **Advanced Topics in Software Engineering**

Date: **22.01.2020**

Outline

1. **Definition of Source code Plagiarism**
2. **Reasons for Source code Plagiarism**
3. **Obfuscation Methods**
4. **Source code Plagiarism Detection & Tools**
5. **Conclusion**
6. **References**

1. Definition of Source code Plagiarism

"Source code plagiarism can be defined as trying to pass off (parts of) source code written by someone else as one's own (i.e., without indicating which parts are copied from which author)" [1]

Some actions leading to source code plagiarism in academia:

- Copying source code
- Converting source code
- Generating source code
- Paying for source code
- Stealing source code
- Exchanging source code

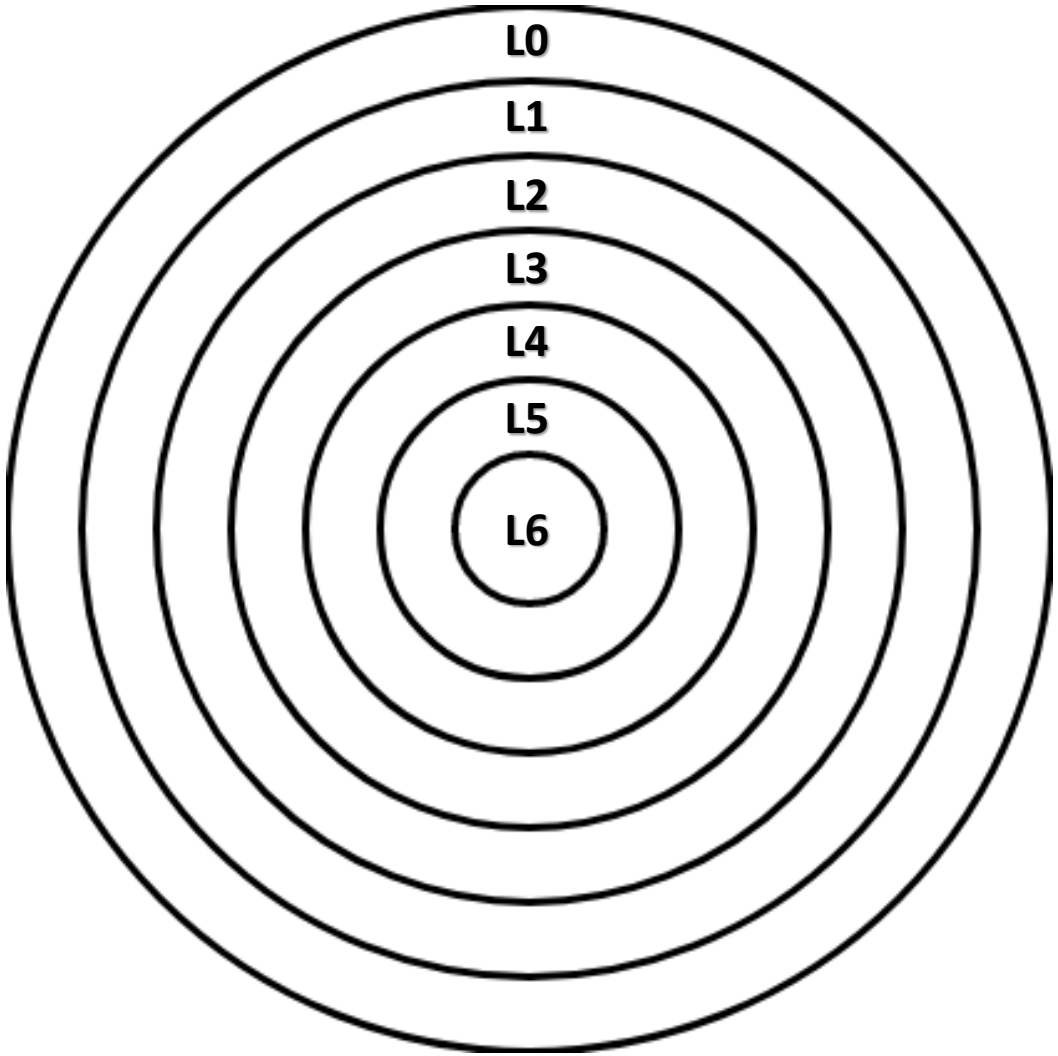
2. Reasons for Source code Plagiarism

- The belief that working in collaboration is acceptable
- The hope that plagiarism will go unnoticed
- The intention of minimizing the work needed
- The availability of many easily reachable resources
- The desire for obtaining higher grades
  - Prove of self-worth
  - Pathological fear of failure
  - Belief for better job options

[2, 3]

Levels of Sophistication of Source code change

- (L0) No changes
- (L1) Changes in comments and indentation
- (L2) Changes in identifiers
- (L3) Changes in declarations
- (L4) Changes in program modules
- (L5) Changes in program statements
- (L6) Changes in decision logic



[6]

3. Obfuscation Methods

Can be divided in 4 main categories:

1. Lexical changes – *"changes which could, in principle, be performed by a text editor. They do not require knowledge of the language sufficient to parse a program."* [5]
2. Structural changes – "requires the sort of knowledge of a program that would be necessary to parse it. It is highly language-dependent." [5]
3. Advanced structural changes – "subcategory of structural changes that require more knowledge of program possibilities and relations between equivalent statements in a specific programming language." [4]
4. Logical changes – "changes that except for structural changes also change the logic (flow) of a program and require certain amount of programming skills and knowledge about the application being developed to be performed correctly." [4]

3.1 Lexical changes

a) “Visual code formatting”; [6]

b) “Comments modification”; [6]

c) “Translation of program parts”; [8]

d) “Modifying program output”; [7]

e) “Identifier rename”; [7]

f) “Changing constant values”; [9]

Lexical changes - Example

ORIGINAL

```
int main()
{
    const int SEED_RANDOMIZER = 7;
    int number, guess;
    number = RNG(SEED_RANDOMIZER);

    printf("Please  enter your number for guess:");
    scanf("%d", &guess);

    /* Check if the number matches user's guess */
    if(number == guess)
        printf("Congratulations, you won the game !\n");
    else
    {
        printf("Wrong guess! Please try a second time:");
        scanf("%d", &guess);
        if(number == guess)
            printf("Congratulations, you won the game !\n");
        else
        {
            printf("Sorry, you lost the game ! ");
            printf("The number was: %d.\n", number);
        }
    }
    return (0);
}
```

PLAGIARIZED

```
int main(){
    const int NIVO_SLUCHAINOST = 12;

    int chislo, dogatka;
    chislo = RNG(NIVO_SLUCHAINOST);

    printf("Моля въведете число за отгатване:");
    scanf("%d", &dogatka);

    /* Проверява дали числото съвпада с докатката */
    if(chislo == dogatka){
        printf("Поздравления, ти спечели играта!\n");
    }
    else{
        printf("Лош опит, пробвай пак:");
        scanf("%d", &dogatka);
        if(chislo == dogatka){
            printf("Поздравления, ти спечели играта!\n");
        }
        else{
            printf("Извинявай, ти загуби играта! ");
            printf("Числото беше: %d.\n", chislo);
        }
    }
    return (0);
}
```

3.2 Structural changes

a) “Reordering independent lines of code”; [7]

b) “Adding redundant lines of code”; [10]

c) “Splitting up lines of code”; [7]

d) “Merging lines of code”: [6]

   i. “Merging lines of code”; [6]

   ii. “Replacing the procedure call by the procedure body”; [11]

Structural changes - Example

ORIGINAL

```
int main()
{
    double ia0, ia1;        /* Coefficients of the input polynomial */
    double p0;              /* P(0) of the output polynomial */
    double ra0, ra1, ra2; /* Coefficients of the output polynomial */

    /* Get the 1st degree input polynomial */
    printf("Enter coefficients from higher to lower order:\n");
    scanf("%lf%lf", &ia1, &ia0);

    /* Get the constant value for the output polynomial */
    printf("Enter p(0) for the output polynomial:\n");
    scanf("%lf", &p0);

    /* Calculate the 2nd degree output polynomial */
    calculate2ndDegree(&ra2, &ra1, &ra0, ia1, ia0, p0);

    printf("Result: %5.3fx^2 + %5.3fx + %5.3f\n", ra2, ra1, ra0);

    return (0);
}
```

PLAGIARIZED

```
int main()
{
    double ra0, ra1, ra2;
    double p0, ia0, ia1;

    /* Get the constant value for the output polynomial */
    printf("Enter p(0) for the output polynomial:\n");
    scanf("%lf", &p0);

    ra0 = p0;

    /* Get the 1st degree input polynomial */
    printf("Enter coefficients from higher to lower order:\n");
    scanf("%lf%lf", &ia1, &ia0);

    ra2 = ia1 * 0.5;
    ra1 = ia0;

    printf("Result: %5.3fx^2 + %5.3fx + %5.3f\n", ra2, ra1, ra0);

    return (0);
}
```

3.3 Advanced structural changes

a) “Changing of statement specification”: [11]

   i. “Changing the operations and operand”; [11]

   ii. “Altering modifiers”; [9]

   iii. “Datatype changes”; [11]

b) “Replacing control structures with equivalents”; [6]

Advanced structural changes - Example

ORIGINAL

```
/* Size of the hard disk defined in the assignment */
const unsigned int DISK_SIZE = 65000;

void GTU_OS::printHardDisk(harddisk &hard)
{
    FILE *fpt = fopen("HardDisk.mem", "w");
    /* According to the assignment each line should contain
    exactly 16 cells of memory in hexadecimal format */
    const int CELL_LIMIT = 16;
    int newLineCounter = 0;

   // Print the first line number (0) as hexadecimal
   fprintf(ftp, "%0004x ", 0);
   for(unsigned int i = 0; i < DISK_SIZE; ++i)
   {
      fprintf(ftp, "%02x ", hard.physicalAt(i));
      ++newLineCounter;
      if(newLineCounter == CELL_LIMIT && i != (DISK_SIZE-1) )
      {
         newLineCounter  = 0;
         fprintf(ftp, "%0004x ", i+1);
      }
   }

   fclose(ftp);
}
```

PLAGIARIZED (?)

```
const unsigned int DISK_SIZE = 65000;

void GTU_OS::print_hard_disk(harddisk &hard)
{
    unsigned int line_counter = 0, i = 0;
    FILE *fpt = fopen("hard_disk.mem", "w");

    while(DISK_SIZE > i)
    {
       if(line_counter == 0)
          fprintf(ftp, "%0004x ", i); // print line num in hex format

       fprintf(ftp, "%02x ", hard.physicalAt(i));
       ++line_counter;

       if(line_counter == 16)
          line_counter = 0; // set new line condition

       ++i;
    }

   fclose(ftp);
}
```

3.4 Logical changes

a) “Simplifying the code”; [10]

b) “Translation of program from other programming language”; [9]

c) “Changing the logic”; [6]

d) “Combining copied and original code”; [11]

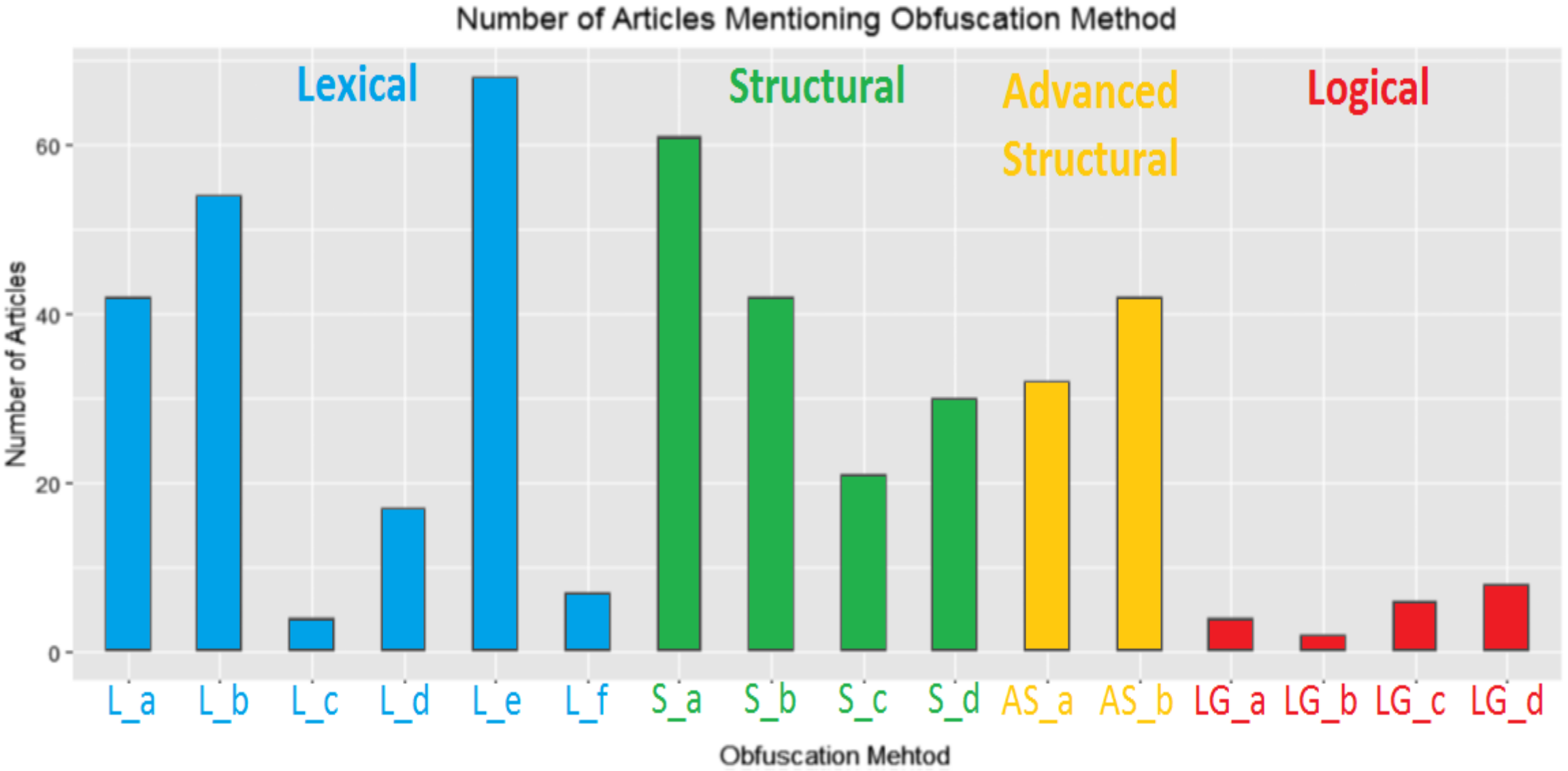
Logical changes - Example

```
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
               swap(&arr[j], &arr[j+1]);
}
```

```
void bubbleSort(int arr[])
{
    int n = arr.length;
    for (int i = 0; i < n-1; i++)
        for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
            {
                // swap arr[j+1] and arr[i]
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
}
```


Number of Articles Mentioning Obfuscation Method
Lexical
Structural
Advanced Structural
Logical
Number of Articles
0
20
40
60
L_a
L_b
L_c
L_d
L_e
L_f
S_a
S_b
S_c
S_d
AS_a
AS_b
LG_a
LG_b
LG_c
LG_d
Obfuscation Mehtod

[4]

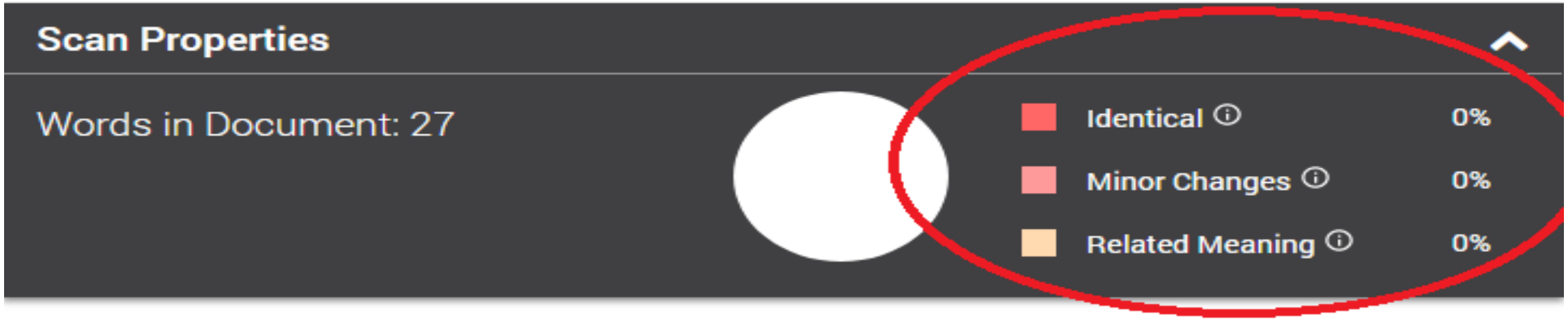
Natural Language Obfuscation Methods (1/4)

“To help teachers, researchers try to find and verify mechanisms for prevention of plagiarism, but also they build tools for automatic similarity detection of potential plagiarism.” [4]

“To help teachers, researchers try to find and verify mechanisms for prevention of plagiarism, but also they build tools for automatic similarity detection of potential plagiarism.” [4]

What is the similarity percentage of the above two sentences in the eyes of a plagiarism detection tool ?

Natural Language Obfuscation Methods (2/4)



File 1

To help teachers, researchers try to find and verify mechanisms for prevention of plagiarism, but also they build tools for automatic similarity detection of potential plagiarism. [4]

File 2

To help teachers, researchers try to find and verify mechanisms for prevention of plagiarism, but also they build tools for automatic similarity detection of potential plagiarism. [4]

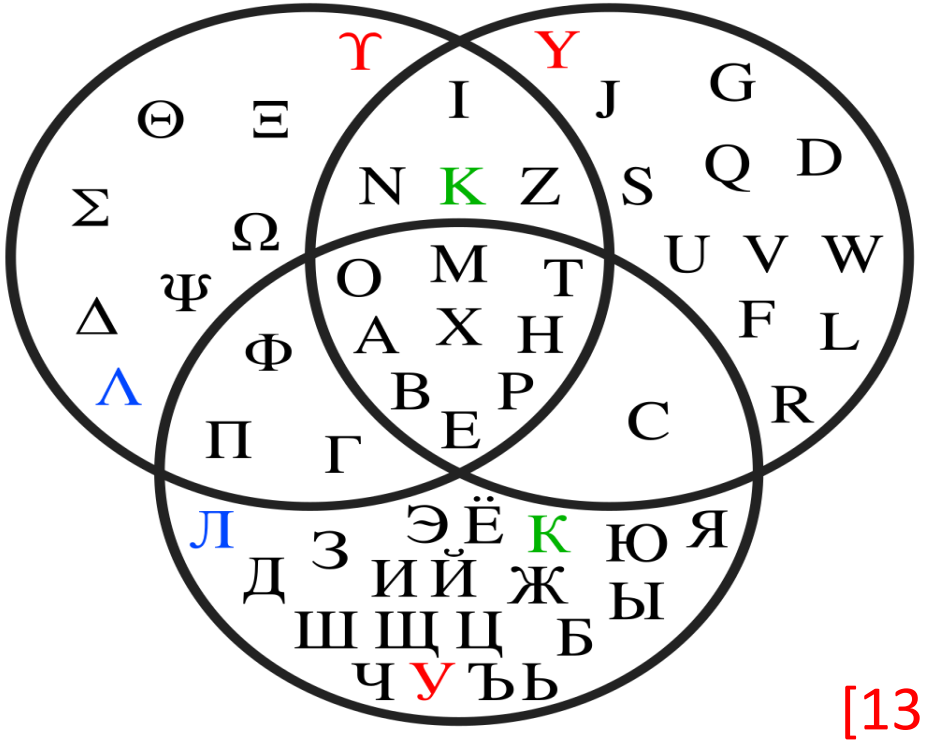
Source: https://copyleaks.com/compare

Natural Language Obfuscation Methods (3/4)

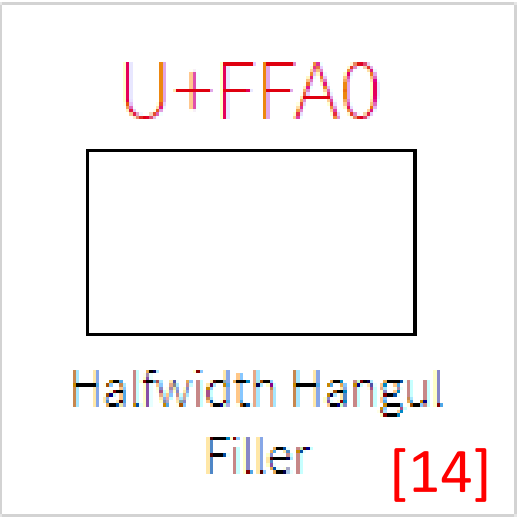
Source: https://www.diffchecker.com/diff

Natural Language Obfuscation Methods (4/4)

1. Changing all whitespace characters with another white font ASCII character – A weak method
2. Changing words with their synonyms – A weak method
3. Changing alphabet letters with similar ones – An average method
4. Using Unicode “invisible” characters at random places – An average method
5. Alter text layer of a pdf – An average method
6. Combination of all methods above – A strong method



[13]



[14]

4. Source code Plagiarism Detection & Tools

Definition: *"A process where someone tries to identify plagiarized source-code regardless the various obfuscation modifications performed on the source-code."* [4]

Donaldson et al. about Tools: *"It is certainly safe to say that neither the detection system described in this paper nor any other detection system will find all occurrences of plagiarism. There is an inherent tradeoff between a highly discriminatory system, which overlooks some instances of cheating, and a less discriminatory one which flags many dissimilar programs."* [7]

Top 5 Tools (1/2)

Tool Name	Last year mentioned in a paper	First year mentioned in a paper	Compared to other tools by # times	Times Used
JPlag	2016	2002	37	5
MOSS	2016	1999	29	9
Sherlock-Warwick	2016	1999	4	4
Plaggie	2016	2006	6	0
SIM-Grune	2014	2010	4	2

Other tools that are compared at least 2 times: Ottenstein, Donnaldson, Accuse, Plague, CCFinder, CodeMatch, Sherlock-Sydney, PMD's CPD, SID, Marble, SIM, and **YAP3**

[4]

Top 5 Tools (2/2)

Tool Name	Number of mentions	Open source	Available GUI	Available Offline	Available from
JPlag	**43**	**YES**	**YES**	**YES**	www.jplag.ipd.kit.edu
MOSS	**38**	**NO**	**YES**	**NO**	www.theory.stanford.edu/~aiken/moss/
Sherlock-Warwick	**9**	**YES**	**YES**	**YES**	www.warwick.ac.uk/iasgroup/software/sherlock
Plaggie	**7**	**YES**	**YES**	**YES**	www.cs.hut.fi/Software/Plaggie
SIM-Grune	**6**	**YES**	**NO**	**YES**	www.dickgrune.com/Programs/similarity_tester

These tools support the following programming languages:

(1) JPlag: Java, C#, C, C++, Scheme and natural language

(2) MOSS: C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, 8086 assembly, HCL2

(3) Sh-War: Java, C and natural language (has no parser, so every programming language can be analyzed)

(4) Plaggie: Java

(5) SIM-Gr: C, Java, Pascal, Modula-2, Lisp, Miranda, and natural language

[4]

5. Conclusion

- Plagiarism detection is a growing research field
- Educational systems need plagiarism detection systems
- Conclusions shouldn’t be made only by results of tools
- In order to reduce plagiarism instructors could:
  - choose assignments that allow several interpretations
  - try to avoid assignments that have general solutions
  - try to change assignments each semester

References

[1] J. Hage, P. Rademaker, and N. van Vugt. 2011. Plagiarism detection for Java: A tool comparison. In Proceedings of the Computer Science Education Research Conference (CSERC'11). 33–46.

[2] Schiller R M. E-Cheating: Electronic Plagiarism. Journal of the American Dietetic Association 2005; 105 (7): 1058-1062.

[3] Bennett R. Factors associated with student plagiarism in a post-1992 university. Assessment & Evaluation in Higher Education 2005, 30(2): 137-162

[4] M. Novak, M. Joy, D. Kermek, Source-code Similarity Detection and Detection Tools Used in Academia: A Systematic Review

[5] M. Joy and M. Luck. 1999. Plagiarism in programming assignments. IEEE Trans. Educ. 42, 2 (1999), 129–133. DOI: https://doi.org/10.1109/13.762946

[6] Faidhi J A W, Robinson S K. An empirical approach for detecting similarity and plagiarism within a university programming environment. Computers and Education, 1987, 11(1): 11-19. DOI:https://doi.org/10.1016/0360-1315(87)90042-X

[7] J. L. Donaldson, A. M. Lancaster, and P. H. Sposato. 1981. A plagiarism detection system. ACM SIGCSE Bull. 13, 1 (1981), 21–25. DOI:https://doi.org/10.1145/953049.800955

References

[8] Z. Đurić and D. Gašević. 2013. A source code similarity system for plagiarism detection. Comput. J. 56, 1 (2013), 70–86. DOI:https://doi.org/10.1093/comjnl/bxs018

[9] L. Prechelt, G. Malpohl, and M. Philippsen. 2002. Finding plagiarisms among a set of programs with JPlag. J. Univ. Comput. Sci. 8, 11 (2002), 1016–1038. DOI:https://doi.org/10.3217/jucs-008-11-1016

[10] S. Grier. 1981. A tool that detects plagiarism in Pascal programs. In *Proceeedings of the 12th SIGCSE Technology Symposium on Computer Science Education (SIGCSE'81)*. ACM Press, New York, NY, 15–20. DOI:https://doi.org/10.1145/800037.800954

[11] G. Whale. 1990. Identification of program similarity in large populations. Comput. J. 33, 2 (1990), 140–146.

[12] C. Arwin and S. M. M. Tahaghoghi. 2006. Plagiarism detection across programming languages. In Proceedings of the 29th Australasian Computer Science Conference, Vol. 48. 277–286.

[13] https://en.wikipedia.org/wiki/Letter_(alphabet) – Last visited 21.01.2020

[14] https://www.compart.com/en/unicode/U+3164 – Last visited 21.01.2020