# Plagiarism Detection in Students' Assignments Written in Natural Language

Ivan Gulis, Daniela Chudá, Juraj Petrík

***Abstract:*** *Testing students' knowledge is crucial aspect for e-learning. However, students are often plagiarising their assignments and using obfuscation techniques to hide plagiarising. In this paper we have proposed and implemented new methods for improved detection of obfuscations into PlaDes plagiarism detection system. We have also modified LCS – substring method for improved detection of similar texts by ability to detect N longest common substrings. Implemented changes significantly improved detection results without significant time penalty.*

***Key words:*** *plagiarism, LCS, longest common substring, PlaDes*

## INTRODUCTION

In e-learning testing of students' knowledge is commonly realized by evaluating home written texts (essays). These essays are written in specific topic to test acquired knowledge during study. These texts should be their own work, however, students tend to plagiarise. There are multiple reasons to plagiarise: time pressure, an uninteresting course, a poor attitude from the teacher [3]. Detection of this kind of cheating can be done by human expert or by using some kind of similarity detection tool. But final decision is always done by human expert. These tools are used only as support for finding suspicious assignments in all submitted works, existing papers, internet sources and so on.

There were multiple studies done to show that plagiarism is serious problem in academic field. For example, survey performed in Swinburne and Monash University shows that 85.4% of Swinburne University students and 69.3% of Monash University students were engaged in academic dishonesty [1]. Another study from Slovak University of Technology in Bratislava shows that 33% of students have ever created plagiarism and 66% of students have ever given their work to someone else to plagiarise it [3].

To make thinks little complicated students who are plagiarising are trying to hide that they are cheating. They are using multiple techniques, called attacks or obfuscations. We can distinguish three main types of attacks – character based, word based and document based attacks.

## METHODS AND TOOLS FOR PLAGIARISM DETECTION

We can divide existing methods for similarity (plagiarisim) detection into four categories: n-gram based methods, dynamic programming based methods, frequency based methods, metadata based methods [5, 7]. Numerous tools, which are implementing these methods are available:

**PlaDes** is desktop application developed at Slovak University of Technology in Bratislava [2]. This tool supports multiple methods for similarity detection such as N-gram, LCS, TF-IDF, metadata and is able to work with common text file formats – doc, docx, pdf, txt. Also it is using stop words and numbers removal, lemmatization, stemming as pre-processing methods.

**Ferret** is application developed at University of Herfordshire [6]. It contains only 3-grams method for similarity detection and supports doc, docx, pdf and txt. It is not doing any pre-processing on tested files.

**WcopyFind** is open-source portable Windows application [4]. According to our testing it is using Greedy-String-Tilling method. and supports multiple languages.

## EXPERIMENT AND OUR METHOD

Aim of the experiment was to benchmark existing tools and methods to find out weaknesses.

Dataset used for this experiment consists of multiple file pairs – original and modified ones (plagiarised, obfuscated), one pair for every obfuscation type. Because majority of tested tools are designed for English language, the files in dataset were written in English (except the situations when we needed samples in Slovak language). Obfuscation types were as follows: exact copy, adding of quotation, synonyms substitution, numbers substitution, words reordering, homoglyphs, spaces substitution by white characters, images instead of text and PDF layers.

**PlaDes** was most successful one amongst all tested tools due to usage of LCS – subsequence similarity detection method, which is able to resist a lot of attack types.

**Ferret** is using 3-grams method for similarity detection, which has serious problems with sophisticated attacks. Also when the document consists exclusively from images, the application was not working at all.

**WCopyFind** has similar results like PlaDes and is relatively resistant to attacks thanks to its unique method to compare and find same substrings, unfortunately this tool does not contain synonyms dictionary.

Based on these findings we have proposed and implemented four methods for detection of chosen attacks into PlaDes tool. Every of these methods is resistant against at least one attack. Some of these methods are even able to detect multiple types of attacks.

**Detection by number of suspicious characters** is focused on eliminating attacks, which are using unknown characters and character replacement. These attacks are: usage of homoglyphs of invalid characters, changing spaces for white characters and PDF layering.

**Detection by character frequency analysis** is also focused on attacks which are using character replacement. We are utilizing histograms of Slovak and English languages and comparing it with histogram of the tested document. We can detect attacks such as usage of homoglyphs of valid characters, space replacement by white characters, using text a visual layers of PDF and text replacement by images.

**Detection by average word length** is designed for detecting space replacement attacks. It is calculating average word length – if this type of attack is used average length of the word is naturally bigger.

**Detection by data volume analysis** is aimed on detecting attacks based on text and visual layer of PDF files and also on attack replacing text by images. We are extracting text from images and after that we are comparing volume of text from images and text from document (normal text).

We have also proposed and implemented into PlaDes tool **modifications for LCS – substring method** - we have extended method by ability to detect N longest common substrings (not only the longest one).

## RESULTS

We have used dataset in Slovak language created by modifying students' assignments. These assignments also contain images and charts.

### Modified and old LCS – substring

We are comparing old and modified LCS – substring method. Old version is working only with the longest substring (N = 1). Modified version is able to work with N longest substring – we have tested N = 10, 100 and 1000 (Table 1).

Table 1 Comparison of modified and old LCS

| Plades (%) | LCS - old (1) | LCS - modified (10) | LCS - modified (100) | LCS - modified (1000) |
|---|---|---|---|---|
| synonyms | 3,67 | 19,77 | 52,37 | 53,61 |
| numbers -> text | 2,14 | 15,72 | 58,45 | 59,31 |
| word reordering | 2,58 | 16,16 | 28,34 | 28,34 |
| homoglyphs i -> L | 1,08 | 8,13 | 37,84 | 41,59 |
| homoglyphs i -> L, cl -> d, rn -> m | 0,45 | 3,47 | 10,63 | 10,63 |
| homoglyphs - cyrillic (50% a,e) | 0,97 | 3,92 | 16,25 | 16,25 |
| homoglyphs - cyrillic (100% a,e) | 0,23 | 1,99 | 2,84 | 2,84 |
| homoglyphs - cyrillic (100% 8 characters) | 0,24 | 0,24 | 0,24 | 0,24 |
| white characters -> spaces (50% L) | 7,1 | 9,68 | 9,68 | 9,68 |
| white characters -> spaces (100% L) | 0,47 | 0,47 | 0,47 | 0,47 |
| PDF layers usage | 0,74 | 1,3 | 1,3 | 1,3 |
| images instead of text | 0 | 0 | 0 | 0 |

Modified LCS has significantly better results than old one – results are 7 to 8 times better with N = 10. Increasing N value above 100 is not worth it. Total time for N = 1000 is approximately two times more than with N = 1.

**Other detection methods**

We used same dataset as in benchmark above. We were comparing modified LCS (with N = 100) with these detection methods too, to show that common detection methods are not good enough for detecting sophisticated plagiarism detection attacks. Table 2 shows yes if the document was marked as suspicious (was obfuscated).

Table 2 Comparison of implemented plagiarism detection methods

| Plades (modified) | LCS - substring (100) | Number of suspicious characters | Character frequency analysis | Average word length | Data volume analysis |
|---|---|---|---|---|---|
| homoglyphs i -> L | 37,84 | no | yes | no | no |
| homoglyphs i -> L, cl -> d, rn -> m | 10,63 | no | yes | no | no |
| homoglyphs - cyrillic (50% a,e) | 16,25 | yes | no | yes | no |
| homoglyphs - cyrillic (100% a,e) | 2,84 | yes | no | yes | no |
| homoglyphs - cyrillic (100% 8 characters) | 0,24 | yes | yes | yes | no |
| white characters -> spaces (50% L) | 9,68 | no | no | yes | no |
| white characters -> spaces (100% L) | 0,47 | no | yes | yes | no |
| PDF layers usage | 1,3 | no | no | no | yes |
| images instead of text (doc) | 0 | no | no | no | yes |
| images instead of text (pdf) | 0 | no | no | no | yes |
| images instead of text (docx) | 0 | no | no | no | yes |

All attacks which were successful (not detected) by modified LCS method are now successfully detected by at least one detection method.

## CONCLUSIONS AND FUTURE WORK

Experiment shows, that existing plagiarism detection systems are not able to detect sophisticated plagiarism detection attacks. Based on this we have proposed and implemented multiple methods – modified LCS, detection by number of suspicious characters, character frequency analysis, average word length and data volume analysis. These changes and new methods were implemented into existing plagiarism detection system PlaDes. With these methods we were able to detect all forms of tested obfuscations with no significant time penalty.

However, there is still room for improvement. Biggest weakness of implemented methods is threshold determination, which needs to be done by expert at the moment and probably could be done automatically in the future. Also OCR used in text extraction from images needs time complexity optimizations.

Another topic for future research is trying to eliminate and detect obfuscations such as document damaging or document locking. But there are also hardly detectable obfuscations - text translating (from one language to another language) and the hardest ones – ideas stealing and also submitting someone else work as their own (the assignment was specifically created for someone - for money for instance) [8].

## REFERENCES

[1] C. Arwin and S. M. M. Tahaghoghi, "Plagiarism Detection across Programming Languages," *Twenty-Ninth Australas. Comput. Sci. Conf.*, p. 10, 2003.

[2] D. Chuda and P. Navrat, "Support for checking plagiarism in e-learning," *Procedia- Soc. Behav. Sci.*, vol. 2, no. 2, pp. 3140–3144, 2010.

[3] D. Chuda, P. Navrat, B. Kovacova, and P. Humay, "The issue of (software) plagiarism: A student view," *IEEE Trans. Educ.*, vol. 55, no. 1, pp. 22–28, 2012.

[4] H. Dreher, "Automatic Conceptual Analysis for Plagiarism Detection," *Issues Informing Sci. Inf. Technol.*, vol. 4, pp. 601–614, 2007.

[5] A. S. Hamza Osman Naomie; Abuobieda, Albaraa, "Survey of Text Plagiarism Detection," *Comput. Eng. Appl. J.*, vol. 1, no. Vol 1, No 1: June 2012, pp. 37–45, 2012.

[6] C. Lyon, R. Barrett, and J. Malcolm, "A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector," *Proc. 1st Int. Plagiarism Conf.*, pp. 1–7, 2004.

[7] M. Mozgovoy, T. Kakkonen, and G. Cosma, "Automatic Student Plagiarism Detection: Future Perspectives," *J. Educ. Comput. Res.*, vol. 43, no. 4, pp. 511–531, 2010.

[8] A. H. Osman, N. Salim, M. S. Binwahlan, R. Alteeb, and A. Abuobieda, "An improved plagiarism detection scheme based on semantic role labeling," *Appl. Soft Comput.*, vol. 12, no. 5, pp. 1493–1502, 2012.

## ABOUT THE AUTHOR

Bc. Ivan Gulis, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Phone: +421 948 976 571, E-mail: xgulisi@stuba.sk.

Assoc. Prof. Mgr. Daniela Chudá, PhD., Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Phone: +421 2 210 22 318, E-mail: daniela.chuda@stuba.sk.

Ing. Juraj Petrík, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Phone: +421 2 210 22 331, E-mail: juraj.petrik@stuba.sk.

**The paper has been reviewed.**