The Similarity Detection in Slovak Texts by Compression Method

Martin Uhlík[*]
Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
uhlik06@student.fiit.stuba.sk

Abstract
This paper deals with similarity and plagiarism with a focus on the Slovak texts. It presents and analyzes standard methods and tools used to detect plagiarism in order to use the conclusions of its own solutions. It explains the principle of dictionary method for data compression known as the Lempel-Ziv, which idea of creating the dictionary is used as the basis for our method proposal to detect plagiarism in texts. Self-designed dictionary method was implemented in the application. In conclusion, achievements are presented, which were compared with results of analyzed tools on a sample of the Slovak texts.

Categories and Subject Descriptors
I.1.2 [**Symbolic and algebraic manipulation**]: Algorithms—*Analysis of algorithms*; I.2.7 [**Artifical intelligence**]: Natural Language Processing—*Text analysis*

Keywords
Similarity, plagiarism, Slovak texts, Lempel-Ziv, dictionary method

1. Introduction
Now, the problem of plagiarism and copying with easy availability of texts related to the development of information technology and the Internet is gaining more and more respect. Especially in the sphere of education, students often resort to plagiarism, which can be unmotivated and may be resulting in degradation of the education. Whether the plagiarism is conscious or unconscious, it is necessary to fight against both.

One of the major problems of plagiarism is determining what plagiarism is and what is not. The most common signs of plagiarism are for example [3]: missing or incorrect references, amount of similarity between texts, long sequences of common text, common spelling mistakes, incoherent text, and others. Standardly, these factors are used as bases for evaluation of similarity of texts, from which we can determine the likelihood that it is a plagiarism or not. The more the plagiarist inserted into the text his own creative work the more difficult is the detection. We can gradually divide levels of plagiarism into: copy the text or its parts, relocation and modification parts of the text, replacing words by their synonyms and taking someone's idea as it was your own.

There have been developed several methods to detect plagiarism with varying success. Each may be appropriate for different levels of plagiarism. Research and development in this area is important and also adaptation of methods to specification of the languages - in our case, Slovak. The aim of this paper is to describe the own method proposal to detect plagiarism in texts and its implementation followed by evaluation results. The main idea of method proposal was to use and adapt the principle of the Lempel-Ziv's method [6], i.e. compression of similar texts (containing the same sequence of text) is more effective than compression different texts. From the character of a method proposal follows that it is suitable for syntax domain of plagiarism detection, specifically the level coping, relocation and modification parts of plagiarized text.

2. Related work
This section describes existing methods and tools for the plagiarism detection in texts. However, before specifying methods, it is necessary to mention the text's pre-processing, which is used to adapt a text for purpose to increase the effectiveness of the method. For each language and each method is appropriate different pre-processing or its combination [1]. Some basic types of text's pre-processing are: Punctuation and/or Diacritical marks removal, Stop-word removal, Lemmatization or Stemming, Synonymy recognition and others.

[*]Master degree study programme in field: Information Systems. Supervisor: Dr. Daniela Chudá, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, STU in Bratislava. Work described in this paper was presented at the 7th Student Research Conference in Informatics and Information Technologies IIT.SRC 2011 and awarded by ACM Slovakia Chapter Award.

Uhlík, M. The Similarity Detection in Slovak Texts by Compression Method. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies, Vol. 3, No. 2 (2011) 122-125

2.1 Methods for plagiarism detection

Below, there are listed methods for plagiarism detection. Their detailed description is in [3, 4]. The methods are:

- N-grams
- Longest Common Subsequence (LSC)
- Term Frequency (TF)
- Inverse Document Frequency (IDF)
- Greedy String Tilling (GST)

2.2 Tools for plagiarism detection

There are many tools to detect plagiarism in texts. Some of them also allow the search of similar texts on the Internet. We chose a few mostly free tools, in which we focus on quality and presentation of results, speed of processing, intuitiveness, the method used and other features that would help us with proposal our own tool. Also, the results of the analyzed tools were important to compare and evaluate our one. For each tool, a basic description is given with the results of our test on the same sample texts.

WCopyFind[1] is developed on the platform .NET of University in Virginia and it is freely available with source code. Its advantage is simplicity and speed. It supports text file formats, HTML and DOC. It allows to the user a lot of pre-processing and detection settings, but it does not use sophisticated methods, e.g. lemmatization. The implemented algorithm is based on matching common substrings and the number of common words. The tool extracts from each document parts of the text, in which it is looking at least word match of specified length. This tool allows you to move documents into a group, in which they are not compared to each other. It prints both the mutual similarities and also offers the option to view side by side. In the test, after appropriate parameters setting, the tool detected 19 of 27 plagiarisms (only 16 in default settings). In another extreme case of settings, a tool included to results also those, which were not plagiarism at all. Test revealed minor deficiencies WCopyFind in finding plagiarism in texts of different sizes, when setting the parameters had to be compromise. It also had problems with the Slovak language. Despite the result, it was also good in regard to speed, when the process took only 7 seconds.

PlaDeS [2] was created by students at our faculty led by Dr. Daniela Chudá. It is also free application built on the platform .NET. It is very clear with an intuitive user interface. PlaDeS supports DOC and PDF formats and allows you to insert entire ZIP archives. The great advantage is the support of Slovak language (lemmatization). The latest version has implemented method of N-grams. Printing the results in different ways is excellent (list, tree and groups). The selected pair of documents can be viewed side by side. In the test, while significantly reducing the marginal similarity on 10% except 1, all possible pairs of plagiarism were revealed. On the other hand, quality results were achieved on the expense of time, when analysis took the order of minutes for the test sample documents. Visual information by the time course analysis was positive.

[1]http://www.plagiarism.phys.virginia.edu/Wsoftware.html

JPlag[2] is client - server application that is implemented in Java. It is designed mainly for comparing program's source codes, but also allows the detection of plagiarism in texts. However, it only supports text file formats (TXT). The tool algorithm uses tokenization and subsequent comparison with GST method (Greedy String Tiling). Analysis takes place on the server side. Results are returned in the form of HTML pages. View in groups of similar documents is very clear. Detailed view side by side we consider to be the best from the tested tools. JPlag had problems with Slovak, which was also reflected in the view of results. In the test, the tool detected 74% plagiarism in default settings (96% after their modification). Total time 20 seconds was also positive.

3. Proposal own method

The main idea is to use method of Lempel-Ziv. Lempel-Ziv method is now mostly used on data compression. Experimentally this method was used on comparison of DNA and also text [5]. Principle of the method is the fact that for the same sequence of text the compression becomes more efficient. From Lempel-Ziv method, we decided to use and adapt the principle of creating a dictionary from which we obtain the parameters for evaluating similarity. The main goal of this proposal is to create a new method that offers a new approach to previously used methods on plagiarism detection in texts.

3.1 Description of creating a dictionary of Lempel-Ziv method

Creating a dictionary is part of the encoding process of Lempel-Ziv method [6]. The dictionary you can imagine as a table where the row contains index (codeword) and the corresponding character string. Dictionary fulfils the following prefix condition: if some string of characters is saved in the dictionary, so its all prefixes are saved there. In Table 1 and Table 2, an example of method is given for the input string I = *abbacbabcbaccbaaccb* and alphabet $A = \{a, b, c\}$.

Encoding steps:

1. Initialize the dictionary to contain all characters an alphabet $\{a, b, c\}$
2. Find the longest string S in the dictionary that matches the current input
3. Emit the dictionary index for S to output and remove S from the input
4. Add S followed by the next symbol in the input to the dictionary
5. If it is something on the input then go to step 2 (else finish).

3.2 Method proposal

Basis of own method are built on description contained in section 3.1. However, there were several key changes:

- whole words are inserted to the dictionary (not just characters)

[2]https://www.ipd.uni-karlsruhe.de/jplag

Table 1: Input string of example and the corresponding encoded output.

Input string	a	b	b	a	c	ba	b	cb	ac	cba	acc	b
Output string	0	1	1	0	2	5	1	7	6	10	11	1

Table 2: Dictionary of compression method example.

Index	Appropriate string of characters (S)	Index	Appropriate string of characters (S)
0	**a**	7	cb
1	**b**	8	bab
2	**c**	9	bc
3	ab	10	cba
4	bb	11	acc
5	ba	12	cbaa
6	ac	13	accb

- the first initialization of the dictionary is not required (the dictionary is initialized by new words continuously, as the input text comes)
- we monitor the count of phrases (called compression) in the dictionary, which is the most important parameter for our method
- we do not deal with encoding of output string

Generally, our own method is based on the creation of the dictionary from the input text, which is divided into words. If the word from an input is not in the dictionary, it is added to dictionary. If this word is in the dictionary, to this word is added the next word from input. This will create a phrase. If the phrase is in the dictionary, to this phrase is added the next word from input in similar way. If the phrase is not in the dictionary, it is added to dictionary and the parameter count of the compressions is incremented. The process is repeated until the end of the input. The whole input text is processed this way.

We have developed two approaches to comparison and acquisition the similarity of two texts:

1. Comparison the second text with a dictionary of first text (dictionary - text)
2. Comparison dictionaries of both texts (dictionary - dictionary)

In the first approach, at the beginning the dictionary is created from the first text. Then this dictionary is used to "compress" the second text, but it is not modified. More specifically, the words and phrases of the second text are compared against the dictionary of the first text. The phrase is made up from word only when this word is in the dictionary. And next word from the input is added to the phrase only when this phrase is in the dictionary. All possible phrases that are found and also are in dictionary, they are counted (this number represents parameter ct_2d_1 in Formula (1)). For the first approach, metrics SM1 (Formula (1)) and SM2 (Formula (2)) have been developed to detect the similarity. It is better when the longer text is processed first.

$$SM1 = (\frac{ct_2d_1}{c_1}) * 100[\%] \qquad (1)$$

- ct_2d_1 - count of compressions of the second text in regard to dictionary from the first text
- c_1 - count of compressions in the dictionary from the first text

The disadvantage metric SM1 is insensitive to differences in the length of compared text and assigning a greater similarity to longer texts. Modifying the previous formula for the ratio of words of texts and approximate function (denominator of the Formula (2)) based on the amount of words positively influences to these adverse impacts. Metric SM2 is the result:

$$SM2 = \frac{SM1 * (w_1/w_2)}{(10^{(1.8976+0.0421*\log_{10}(w_1+w_2))})/100}[\%] \qquad (2)$$

- w_1 (w_2) - amount of words of first (second) text

In the second approach, one dictionary is created for the first text and other dictionary is created for the second text. Then both dictionaries are compared to each other. We are matching and counting exact common phrases (not single words). The resulting number represents parameter sc in the Formula (3), which is a metric for this second approach.

$$SM3 = \frac{sc}{min\{c_1, c_2\}} * 100[\%] \qquad (3)$$

- sc - amount of common compressions in dictionaries from the first and second text
- c_1 (c_2) - amount of compressions in the dictionary from the first (second) text

4. Application and experimental results

The application has been implemented in order to verify the own method and comparison its results with other tools on a sample of the Slovak texts. These texts are from student's essays, thesis and technical texts from the Internet. They have from 504 to 8642 words. Together there are 25 texts, respectively 27 unique all possible pairs of plagiarism. They have different similarity and different levels of plagiarism. Basic requirements for the application were: loading and processing N input text files, choice

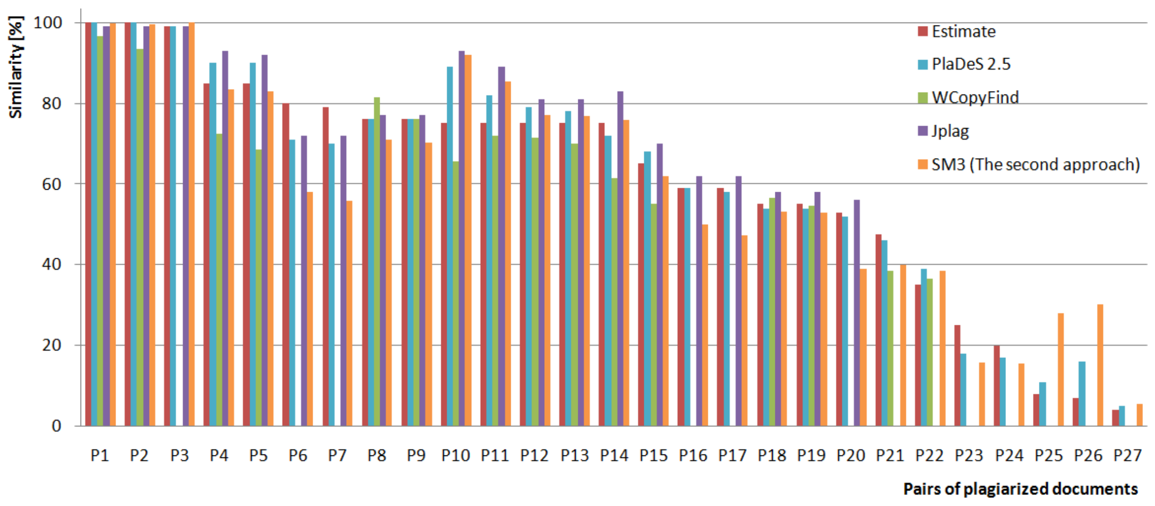

Figure 1: Comparing the results of tools and our application with metric SM3 on sample texts.

any metrics of both approaches on comparison texts, and presentation of results as a percentage similarity of pairs of texts. It also allows different types of pre-processing and their combinations. For our method, removing stop words proved to be the best. In the graph (see Figure 1), the results of own method are given for metric SM3 also using removal stop words. SM3 metric gives more accurate results of similarity than SM2. However, SM2 detects more similarities in the texts, but it also has a higher threshold of similarity. Overall, we consider the metric SM3 for the best on the basis of the achievements. Estimate of similarity was calculated on the basis of amount of words in common parts of texts.

Evaluation own method (metric SM3):

- analysis and evaluation process is fast (0.6 sec. for the sample texts - it is the best time from all tested tools) and method may be implemented as part of pre-processing
- resistance to displacement parts of the text
- in test method detected all 27/27 plagiarism pairs
- results of own method approximately reflect the results of the analysis tools and they are visibly better than the tool WCopyFind
- worse detection similarity of short texts (in less than 500 words)
- method would fail (identify zero similarity) in short text, which contained no word repetition
- method is also suitable for other languages

5. Conclusions and future work

In this paper, new method has been described with the metrics on detection of similarity in pairs of texts. It is based on the principle of compression method Lempel-Ziv. The idea is creating dictionary from text, comparison and acquisition of parameters from created dictionary. Comparison of the results indicated a great competitiveness of our method against the existing tools. We plan to use more additional parameters from the dictionary, what can bring even better results. Also, next improvements of method are not excluded.

Furthermore, we want to use the dictionary to get the characteristic phrases of the text for searching its similar texts on the Internet. Also, we want to assign texts to groups and create a user interface appropriate to presentation the results of similarity that the user himself can decide what is and what is not plagiarism.

Acknowledgement. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0971/11 and by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 345-032STU-4/2010.

References

[1] Z. Ceska and C. Fox. The influence of text pre-processing on plagiarism detection. In *Proceedings of the International Conference RANLP-2009*, pages 55–59. Association for Computational Linguistics, Borovets, Bulgaria, 2009.

[2] D. Chudá and P. Návrat. Support for checking plagiarism in e-learning. In *Procedia - Social and Behavioral Sciences, Innovation and Creativity in Education*, volume 2, pages 3140–3144, 2010. ISSN 1877-0428.

[3] P. Clough. Old and new challenges in automatic plagiarism detection. *National UK Plagiarism Advisory Service (Online)*, 2003.

[4] K. Papineni. Why inverse document frequency? In *NAACL '01 Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Pittsburgh, Pennsylvania, 2001.

[5] L. Seaward and S. Matwin. Intrinsic plagiarism detection using complexity analysis. University of Ottawa, 2096 Madrid Avenue, Ottawa, ON,K2J 0K4, 2009.

[6] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978. ISSN 0018-9448.