Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 1329–1338

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

A Detection Method for Plagiarism Reports of Students

Daisuke Sakamoto[a*], Kazuhiko Tsuda[b]

[a]*Honda Motor Co., Ltd., 8-1, Hon-machi, Wako-shi, Saitama-Pref, 351-0114, Japan*
[b]*University of Tsukuba, 3-29-1, Otsuka, Bunkyo-ku, Tokyo, 112-0012, Japan*

Abstract

In recent years, plagiarism that uses the sentences or phrases of others without permission has become a social problem. It is widely spread from very familiar student reports to novels and worldwide academic papers.

In this paper, we deal with plagiarism in student reports, and explain the plagiarism patterns often found there. Then we propose a method to detect them efficiently and accurately.

This method is based on the way of making two texts to be compared with into one-dimension string respectively, repeating shift and mutual comparison, and checking the matching section of words. This method can accurately detect perfect matches of any size, regardless of its placement in the text.

To this basic detection method, we combine some heuristics which are estimation of detection possibility and compression of strings, to improve both detection accuracy of the plagiarism sections and the reduction of calculation time, and propose it as a plagiarism detection method.

This method has only one parameter for plagiarism judgement, and it is also intuitive and easy to set.

Finally, we show the effectiveness of the proposed method, especially the introduced heuristics, using real data.

Keywords: Pragiarism, Heuristics

1. Introduction

The spread of communication means, including the Internet, has made it possible to browse many texts at any time and place. In addition, it is possible to easily steal other people's sentences or phrases without permission since copy and paste is a standard function of PCs. At the same time, the guilt of plagiarism is also decreasing due to its

* Corresponding author. *E-mail address:* daisuke_sakamoto@hm.honda.co.jp

simplicity. As you know, this plagiarism has spread widely from familiar student reports [1] to novels and global academic papers, and is becoming a social problem.

This paper focuses on student reports among these plagiarism issues.

Some students often plagiarize from reports written on similar reporting assignment in the past. Therefore, the brute force calculation of past and present reports is necessary to find plagiarism. In addition, it is obvious that visual inspections will be impossible because the past reports increase year by year, and also the number of students who submit the reports is many.

On the other hand, considering that students are taking same lectures and being given same reporting assignment, they may use many similar words even if they don't plagiarize from others on purpose. That is, there is a feature that it is difficult to determine whether it is plagiarism only by extracting and comparing sharing words.

In consideration of these situations, this paper aims to provide a method to detect student report specific plagiarism with high accuracy and efficiency.

In this paper, we first define about "plagiarism" which is to be detected. Basically, that has obvious copy and paste traces.

Next, we will explain plagiarism patterns commonly found in student reports. There are various patterns of plagiarism from highly malicious ones such as a complete copy, to small parts plagiarism and the like.

Next, we propose a method for detecting plagiarized sections to deal with these various plagiarism patterns. The proposed method is composed of a simple string matching method and some heuristics to support it. These heuristics enable to improve detection accuracy and reducing calculation time.

Finally, we show the effect of the proposed method, especially the effect of heuristics, using real data.

2. Related Works

There are many different ways to compare two texts and detect plagiarism [2][3].

TFIDF [4] and Cosine Similarity [5][6] are methods as treating a text as a set of words. They basically don't take account of word order since based on set theory. Thus, the more shared words used between comparison texts, the more similar they are considered. This feature can lead to false detections of plagiarism depending on the reporting assignment. For example, when the reporting assignment is about a celebrity, many people already have a common impression, and naturally the expression way, that is, the words used, will become similar. As a result, that may be treated as plagiarism report even if there was no intention to plagiarize.

On the other hand, there are also N-gram based methods [7][8][9] that take account of the order of letters or words. This is basically based on detecting a matching section of strings, and trial & error is required to set an appropriate N. In addition, a large amount of calculation is necessary because of iterative partial perfect matching process.

There are also methods of detecting similarity using editing cost [10], but these also need a large amount of calculation basically. In addition, when comparing long texts, it is necessary to devise such as string division, but setting of the division size is also one of the problems.

There are special approaches do not use references of comparison [11]. However, it is usual to show evidence when plagiarism is detected, and references is necessary at that time.

3. Proposal of Plagiarism Detection Method

In this chapter, we will clarify the definition and plagiarism patterns in our task, and propose a detection method.

3.1. Our Target Plagiarism

3.1.1. Definition of Plagiarism

Our goal is to detect plagiarism in student reports. There is a wide variety, from partial plagiarism to malicious full copy. Therefore, we decide that when we compare two texts, if the section of words defined below is confirmed, we consider either of the two to be plagiarizing that section.

- The same words are used in the same order, and the state continues more than a predetermined length in the text.

It is important to mention the word order here. That is, just because the words used are the same, it's not enough to be considered it as a plagiarism. The reason is that, as explained in Chapters 1 and 2, we focus on student reports, and the words used may be similar depending on the reporting assignment.

Also, if the plagiarism is determined based on the word sharing rate, it is generally difficult to estimate an appropriate threshold for determining "plagiarism" in advance. For example, in case a plagiarism text is made using small parts from several different texts, each word sharing rate will be low between plagiarism text and other plagiarized texts.

Hereafter, we call the part of text that meets the above condition as "plagiarism section".

3.1.2. Target Plagiarism Patterns

We will focus on the detection of plagiarism patterns as follows often found in student reports.

- Pattern A: Copy all text (full copy).
- Pattern B: Copy all the text once and modify partially (usually several places) such as exchange, insertion and deletion of words and symbols in the sentence to try to escape from the discovery of plagiarism.
- Pattern C: Copy parts of texts from a specific or multiple documents, and combine them with the plagiarizer's original sentences. In some cases, exchange the placement.
- Pattern D: Hybrid of the above B and C.

An example of Pattern B is shown in Table 1.

After copying a text, this plagiarizer has exchanged the word "baseball" to "football", added adjective “both”, and eliminated double quotations.

Needless to say, the most vicious full copy is the extreme form of this pattern.

Table 1. Example of Plagiarism Pattern B

Plagiarizer Text	I have experienced **both** the **joy** of achieving my goals and the **chagrin** of losing games, through **football** that I have been doing since I was an elementary school student.
Plagiarized Text	I have experienced the **“joy”** of achieving my goals and the **“chagrin”** of losing games, through **baseball** that I have been doing since I was an elementary school student.

An actual example of plagiarism pattern C is too large for this paper to show, so it is shown in Fig. 1 as conceptual model. Here, Text A, B and C mean be plagiarized texts, and Text P1, P2 and P3 mean generated texts by plagiarism. In addition, the strings expressed by x, y, p, q are plagiarism sections and the hyphens are non-plagiarism sections.

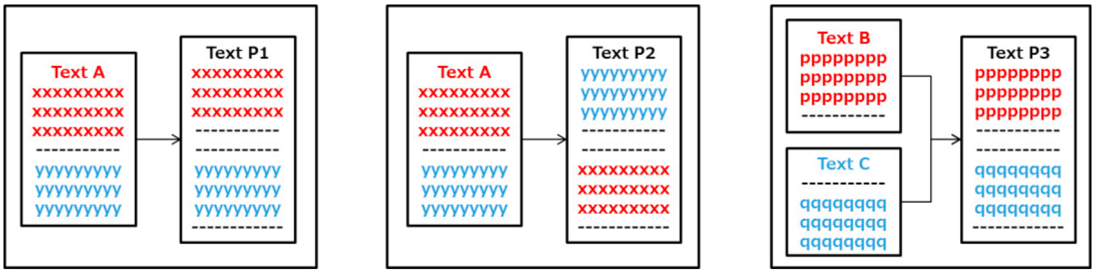


Fig. 1. Examples of Plagiarism Pattern C

3.2. Basic Detection Method of Plagiarism

The detection of the plagiarism sections is to find a matched words section in arbitrary size, between two texts to be compared.

This process requires more calculation than searching for the presence of a pre-specified string. The most steady and easy detection method for our target plagiarism patterns is to make two texts into one-dimensional strings respectively, and repeat string shift and character comparison between them.

An example of the process is shown in Fig. 2. The alphabet in the figure may be a letter, or may be a word after morphological decomposition. It is possible to detect plagiarism which was defined in 3.1.1 with this method regardless of their placement. In this example, "ABC" and "abc" are detected as plagiarism sections. We adopt this as a basic detection method.

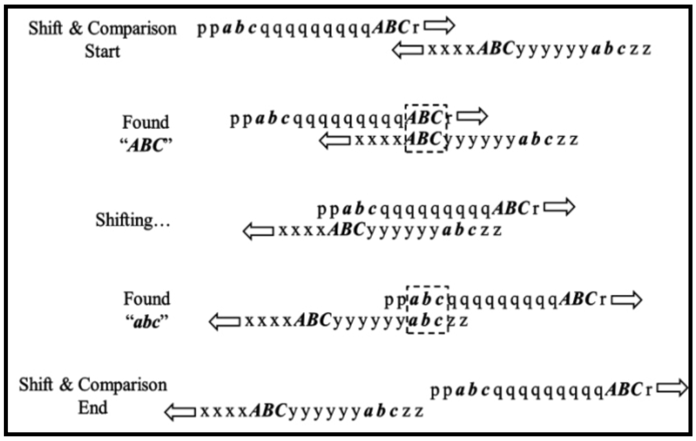


Fig. 2. Process of Basic Detection Method of Plagiarism Section

In this method, it is necessary to determine in advance the threshold of matching string size for detection, but easier to set intuitively as compared with the method using Cosine Similarity described the above. And also, there is an advantage that the process itself is very easy. On the other hand, there is a disadvantage that comparison between large amounts of texts becomes difficult because the amount of calculation for comparison is large. And also, this method may be deceived by simple way like words exchange and fails to detection, because it is based on perfect matching.

Therefore, we try to develop heuristics to improve this basic detection method without reducing its advantage, and combine them as a detection method to propose.

3.3. Detection Method of Plagiarism

The following points are important to improve the above basic detection method.

- Estimate the possibility to detect plagiarized sections with light load calculation, and if there is no possibility, skip calculation for detail comparison, and move to the next texts comparison.
- Shorten the text being compared without losing important information.
- Improve detection robustness against fluctuation of texts.

We show a process of plagiarism detection that uses heuristics to achieve the above points in Fig. 3.

3.3.1. Generation of Noun Arrays from Original Texts

The features of sentences can be roughly determined by nouns. So, we first do morphological decomposition to each text belonging to a group of target texts (texts of inspection target) and reference texts (texts which may be plagiarized), and generate noun only arrays from them in this process. We name the noun-arrayed target texts set as (***Nt***) and the noun-arrayed reference texts set as (***Nr***), and the elements of each set are as (***nt***, ***nr***). From here we simply call ***nt*** the target, and ***nr*** the reference.

In the case of Japanese, this process can reduce the amount of calculation to less than half because it is expected to reduce the string size to about 40%. In addition, there is an advantage that noun array is not naturally influenced by fluctuation of the other part-of-speech except for nouns. This helps to counter part of plagiarism pattern B (See 3.2.1).

3.3.2. Extraction of Shared Words between Target and Reference

In this process, we extract shared words between noun arrays ***nt*** and ***nr***, generated in the previous process, and name the set of them as (***C***).

3.3.3. Calculation of Used Amount of Shared Words in Target and Reference

In this process, we calculate used amount of shared words in the set ***C*** within noun arrays ***nt*** and ***nr*** using equation (1), and name them as *mt* and *mr* respectively. Here, nt_k is an each element of the array ***nt***, and *N* is array size of ***nt***.

$$mt = \sum_{k=1}^{N} f\left(nt_k, \boldsymbol{C}\right)$$
$$f\left(nt_k, \boldsymbol{C}\right) = \begin{cases} \text{String Length of } nt_k & \left(nt_k \in \boldsymbol{C}\right) \\ 0 & \left(nt_k \notin \boldsymbol{C}\right) \end{cases} \tag{1}$$

Of course, ***C*** is the set of shared words, so all its elements are definitely present in ***nt***. Also, the value of *mt* will be equal to or greater than the total string length of all elements in ***C*** because the shared words may be used multiple times in ***nt***. *mr* is calculated by changing nt_k and *N* for ***nr*** in equation (1) as well.

After calculations of *mt and mr*, we have to introduce numeric parameter which corresponds to "predetermined length" in our definition of plagiarism (See 3.1.1), as a threshold to determine whether plagiarism or not.

Here, if *mt* or *mr* which was calculated without considering the word order, is smaller than *M*, it is obvious that plagiarism we defined doesn't exist. Therefore, it is meaningless any more to compare selected ***nt*** and ***nr*** this time, and we take "Skip Route 1" in Fig. 3 to omit following process.

On the other hand, if *mt* and *mr* are larger than *M*, there is still possibility that the plagiarism exists, so we move to next process (3.3.4).

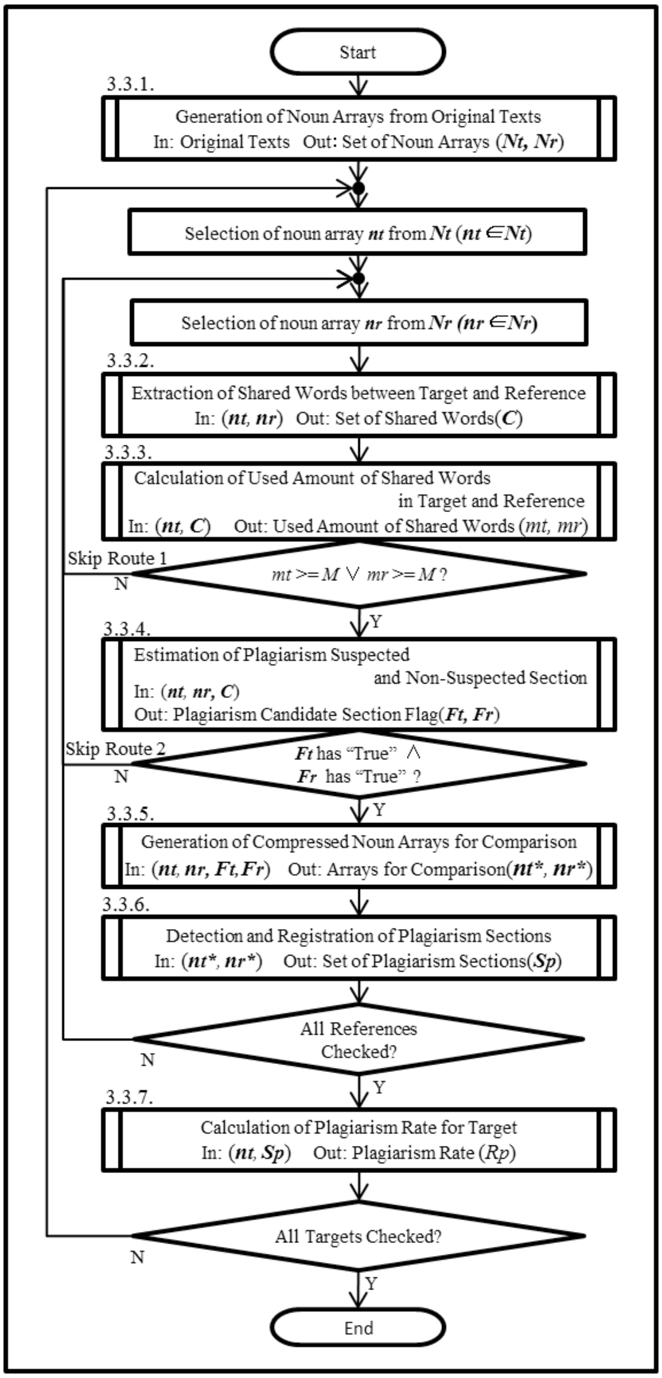

Start
3.3.1.
Generation of Noun Arrays from Original Texts
In: Original Texts Out: Set of Noun Arrays (Nt, Nr)
Selection of noun array nt from Nt (nt ∈Nt)
Selection of noun array nr from Nr (nr ∈Nr)
3.3.2.
Extraction of Shared Words between Target and Reference
In: (nt, nr) Out: Set of Shared Words(C)
3.3.3.
Calculation of Used Amount of Shared Words in Target and Reference
In: (nt, C) Out: Used Amount of Shared Words (mt, mr)
Skip Route 1
N
mt >= M ∨ mr >= M ?
Y
3.3.4.
Estimation of Plagiarism Suspected and Non-Suspected Section
In: (nt, nr, C)
Out: Plagiarism Candidate Section Flag(Ft, Fr)
Skip Route 2
N
Ft has "True" ∧ Fr has "True" ?
Y
3.3.5.
Generation of Compressed Noun Arrays for Comparison
In: (nt, nr, Ft, Fr) Out: Arrays for Comparison(nt*, nr*)
3.3.6.
Detection and Registration of Plagiarism Sections
In: (nt*, nr*) Out: Set of Plagiarism Sections(Sp)
All References Checked?
N
Y
3.3.7.
Calculation of Plagiarism Rate for Target
In: (nt, Sp) Out: Plagiarism Rate (Rp)
All Targets Checked?
N
Y
End

Fig. 3. Process of Detection Method of Plagiarism Section

3.3.4. Estimation of Plagiarism Suspected and Non-Suspected Section

In this process, we decide plagiarism suspected section and non-suspected section using ***nt***, ***nr*** and ***C***. The non-suspected sections on ***nt*** and ***nr*** are compressed in later process.

Step 1: We generate flag arrays ***Ft*** and ***Fr*** having the same length as ***nt*** and ***nr*** respectively, and name them as "plagiarism-suspected flag". If the element of ***nt*** or ***nr*** is also an element of ***C***, the corresponding index element is set "True", otherwise "False" respectively. That is, the shared word locations on ***Ft*** and ***Fr*** are "True". The indexes which were set "True" in this process become plagiarism suspected section for the time being.

Here, we show an example of this process with the array tables of Step 1 in Fig. 4. The ***nt*** and ***nr*** lines in the table indicate the elements of the noun arrays, and we assume that they are composed of the nouns set {p, q, r, s, A, B, C, D, E, F, G, H, I, J} and set { w, x, y, z, A, B, C, D, E, F, G, H, I, J } respectively. Therefore, set ***C*** is {A, B, C, D, E, F, G, H, I, J}, and elements of corresponding index of ***Ft*** and ***Fr*** are set "True".

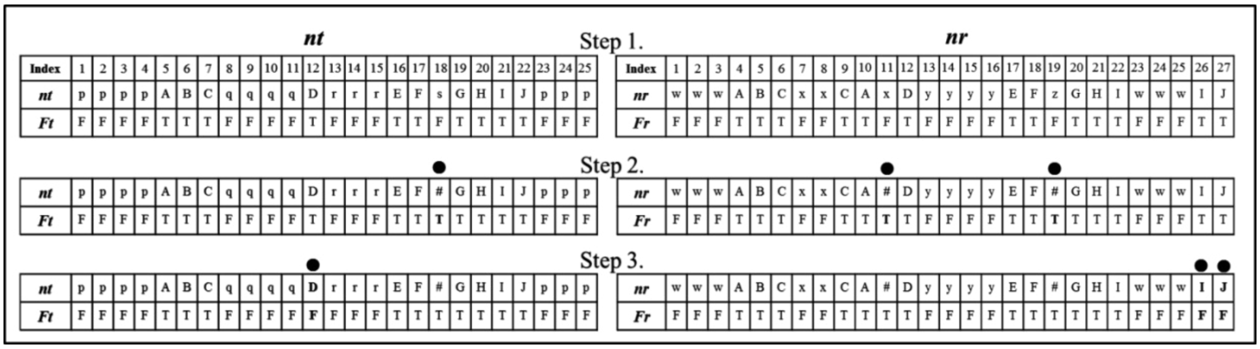
Step 2: We convert the isolated "False" part to "True". The isolated "False" means a point, at which the both front and back are "True" on the flag arrays ***Ft*** and ***Fr***. At the same time, the elements at the same index on the corresponding arrays ***nt*** and ***nr*** are changed to wildcards (#) respectively. The aim of this operation is to counter the camouflage that makes the plagiarism section be shown shorter by dividing longer section using swap of word and so on (See 3.1.2 Pattern B).

Here, we show an example of this process with the array tables of Step 2 in Fig. 4. Since the isolated "False" of ***Ft*** is at Index 18, so is changed to "True", and the element of ***nt*** of the same index is changed from "s" to "#". In ***Fr*** and ***nr***, Index 11 and Index 19 are changed (Dot portions of the table) as well.

Step 3: We convert the small "True" sections to "False". A small "True" section means that elements of the plagiarism-suspected flag array is continuously "True", and also total string length of elements in noun array is less than M in the same "True" indexes. By setting these sections as "False", the compression ratio in the next process becomes high, and as a result, the amount of calculation can be reduced. Also, if all the elements of ***Ft*** or ***Fr*** are "False" in this process, it is obvious that there is no prospect to detect plagiarism section, and we can take "Skip Route 2" in Fig. 3 to omit following process.

Here, we show an example of this process with the array tables of Step 3 in Fig. 4. In this example, it is assumed that the total string length reaches M in case more than two continuous "True" in plagiarism-suspected flag. In this case, the small "True" section in ***nt*** is Index12, and ones in ***nr*** are Index26 and Index27. And corresponding elements of ***Ft*** and ***Fr*** are changed to "False" (Dot portions of the table).

The "True" sections in ***Ft*** and ***Fr*** generated up to this Step 3 will be the plagiarism-suspected sections.



Step 1.

nt

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
nt	p	p	p	p	A	B	C	q	q	q	q	D	r	r	r	E	F	s	G	H	I	J	p	p	p
Ft	F	F	F	F	T	T	T	F	F	F	F	T	F	F	F	T	T	F	T	T	T	T	F	F	F

nr

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
nr	w	w	w	A	B	C	x	x	C	A	x	D	y	y	y	y	E	F	z	G	H	I	w	w	w	I	J
Fr	F	F	F	T	T	T	F	F	T	T	F	T	F	F	F	F	T	T	F	T	T	T	F	F	F	T	T

Step 2.

																		●							
nt	p	p	p	p	A	B	C	q	q	q	q	D	r	r	r	E	F	#	G	H	I	J	p	p	p
Ft	F	F	F	F	T	T	T	F	F	F	F	T	F	F	F	T	T	**T**	T	T	T	T	F	F	F

											●								●								
nr	w	w	w	A	B	C	x	x	C	A	#	D	y	y	y	y	E	F	#	G	H	I	w	w	w	I	J
Fr	F	F	F	T	T	T	F	F	T	T	**T**	T	F	F	F	F	T	T	**T**	T	T	T	F	F	F	T	T

Step 3.

												●													
nt	p	p	p	p	A	B	C	q	q	q	q	**D**	r	r	r	E	F	#	G	H	I	J	p	p	p
Ft	F	F	F	F	T	T	T	F	F	F	F	**F**	F	F	F	T	T	T	T	T	T	T	F	F	F

																										●	●
nr	w	w	w	A	B	C	x	x	C	A	#	D	y	y	y	y	E	F	#	G	H	I	w	w	w	**I**	**J**
Fr	F	F	F	T	T	T	F	F	T	T	T	T	F	F	F	F	T	T	T	T	T	T	F	F	F	**F**	**F**

Fig. 4. Example of Estimation Process for Plagiarism Suspected Sections

3.3.5. Generation of Compressed Noun Arrays for Comparison

In this process, we generate compressed noun arrays for comparison. The arrays ***nt*** and ***nr*** are compressed and converted according to ***Ft*** and ***Fr*** respectively. The elements of ***nt*** and ***nr*** are left if the corresponding elements of ***Ft*** and ***Fr*** are "True" respectively, because they are plagiarism-suspected section. On the other hand, the elements of ***nt*** and ***nr*** are compressed into one element (space) if the corresponding elements of ***Ft*** and ***Fr*** are "False" respectively, because they are not plagiarism-suspected section. We name these compressed noun arrays as ***nt**** and ***nr**** respectively.

Here, we show an example of this process in Fig. 5. The original array size of ***nt*** and ***nr*** were 25 and 27 respectively, but after compression to ***nt**** and ***nr****, the size were 11 and 15 respectively. This means that the amount of calculation for comparison is reduced to about 50%.

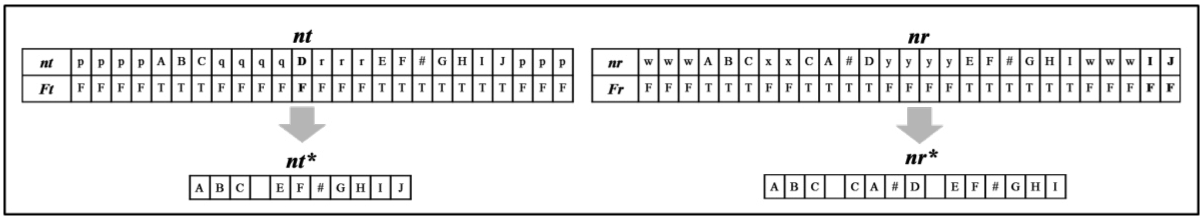

nt
nt p p p p A B C q q q q D r r r E F # G H I J p p p
Ft F F F F T T T F F F F F F F F T T T T T T T F F F
nt*
A B C E F # G H I J
nr
nr w w w A B C x x C A # D y y y y E F # G H I w w w I J
Fr F F F T T T F F T T T T F F F F T T T T T T F F F F F
nr*
A B C C A # D E F # G H I

Fig. 5. Example of Noun Array Compression

3.3.6. Detection and Registration of Plagiarism Sections

In this process we apply the basic detection method (See 3.2) to the compressed array ***nt**** and ***nr****. The combination of ***nt*** and ***nr*** that are obviously no prospect to detect plagiarism section in the above process, does not reach this process. Also, even if it is reached, ***nt*** and ***nr*** have been compressed into ***nt**** and ***nr**** respectively, so significant reduction in calculation time can be expected.

We shift one of the ***nt**** and ***nr**** by one index at a time, and try to detect the matching section of words as we showed in Fig. 2. For matching process, the wildcard (#) which introduced in 3.3.4 is considered to match any word. If matching detected, continuous matching word sections are treated as one set, and the string length of each set is calculated separately. That is, there is a possibility that two or more sets of calculations may be done per one comparison of ***nt**** and ***nr****. And if the calculated value is M or more, the set (partial array) is determined to be "plagiarism section" and registered.

For example, in case the comparison between ***nt**** and ***nr**** in Fig. 5, [A, B, C] and [E, F, #, G, H, I] are the registration candidates. If each of these sets has a string length of M or more, it is registered as a "plagiarism section" (however, the length of # is calculated as 0).

Here, we name a set of registered "plagiarism section" as ***Sp***, and its elements as ***sp*** which are noun arrays.

3.3.7. Calculation of Plagiarism Rate for Target

In this process, we calculate the plagiarism rate of ***nt*** using itself and ***Sp*** which is plagiarism section set. Off course, All elements ***sp*** in ***Sp*** are partial arrays of ***nt***. Assuming that the total string length of the array elements in nt, that did not match any array ***sp***, is lo, and the total string length of all elements in ***nt*** is l, the plagiarism rate of ***nt*** is given by equation (2) .

$$Rp = \frac{l - lo}{l} \qquad (2)$$

The plagiarism rate can be used as an indicator when deciding whether to reject the report. For example, it is possible to eliminate reports that are not worth reading automatically by deciding that the plagiarism rate is 50% or more as a rejection.

4. Effect of Proposal Method

In this chapter, we confirm the effectiveness of the proposed method on the plagiarism detection accuracy and calculation time.

We show the conditions of data used for this test in Table 2. We compare with past reports (2017, 18 years) to detect plagiarism included in the 2019's report. And the total number of report comparisons is 22,331,160 (=3915*5704).

In addition, the value of the parameter M used when determining the plagiarism is 30. That is, in case string length of consecutive matching sections is 30 or more, is basically treated as plagiarism.

Table 2. Test Data Conditions

	Inspection Target Reports（2019）	Reference Reports（2017,18）
Number of Reports	3915	5704
Average String Length per Report	499.095	445.845
Language	Japanese	

Since the proposed method is composed of several heuristics, we prepare for some test cases shown in Table 3 to clarify the effects of each. We add the heuristics explained in the process from 3.3.1 to 3.3.5 in sequence. "Case 3" is the case where the proposed heuristics are fully used. In this test, effect of the each heuristic is relatively evaluated based on the result of Case 0, which uses basic detection method (See 3.2) and noun arrays generated from the original text.

Table 3. Test Cases

	Process No.	Case 0	Case 1	Case 2	Case 3
Noun array generation from original text	3.1.1	apply	apply	apply	apply
Estimation of plagiarism detection possibility using shared words (***C***), and taking Skip Route 1 if possible	3.3.3	-	apply	apply	apply
Estimation of plagiarism suspected section using shared words(***C***), and taking Skip Route 2 if possible	3.3.4	-	-	apply	apply
Noun array compression using ***Ft*** and ***Fr***	3.3.5	-	-	-	apply

4.1. Effects for Plagiarism Detection Accuracy

We show the test results about plagiarism detection in Table 4.

There was no difference between Case 0 and 1 about detection performance regarding the presence or absence of plagiarism, and 208 reports were confirmed that had plagiarism sections. On the other hand, the number of detection increased by nine in Case 2 and 3, and moreover, it was confirmed that the average value of the plagiarism rate of reports judged to have plagiarism section also increased by about 0.8%.

Table 4. Comparison of Plagiarism Detection

	Case 0	Case 1	Case 2	Case 3
Number of Plagiarism Reports	208 Counts		217 Counts	
Average Plagiarism Rate	0.5170		0.5254	

From this result, we confirmed that the heuristic added in process 3.3.4 works in the direction of improving detection accuracy. The improvement is due to the fact that the replacement process to wildcards (#) has effectively acted on the camouflage of the plagiarism in which "pattern B" (See 3.3.1).

The division of the plagiarized section using small process like word exchange makes the individual plagiarized section smaller. However, divided sections are joined using wildcards and treated as a large plagiarized section in matching process, which makes more difficult to escape from the detection.

The breakdown of plagiarism rate about 217 cases, which were detected plagiarism, is as shown in Fig. 6. Boldly, the perfect copy was the largest.

4.2. Effects for Calculation Time Reduction

We show the results about the effect of calculation time reduction in Fig.7. The Case 2 had the largest reduction effect in this test. This means the method of skipping the subsequent process using "Skip Route 2", when the size of suspected plagiarism section is small, is effective.

On the other hand, improvement from Case 2 to Case 3 was not so large relatively. That is because the string compression for comparison in 3.3.5 is a method that is used only when there is plagiarism suspected section. So its effect may be more noticeable in cases where the frequency of plagiarism is many.

The proposed method (Case 3) was programmed with "R", and the actual calculation time was about 34 minutes on EC2 in AWS Service (m5.xlarge, AMD64).

From the above results, we confirmed that our proposed heuristic can significantly reduce the calculation time, and improve detection accuracy. Moreover, confirmed that has a practical performance as a plagiarism detection method for student reports.

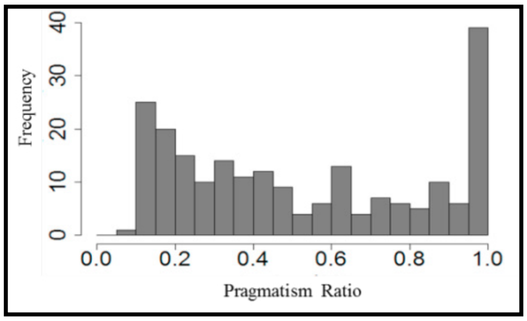


Fig. 6. Breakdown of Pragmatism Rate

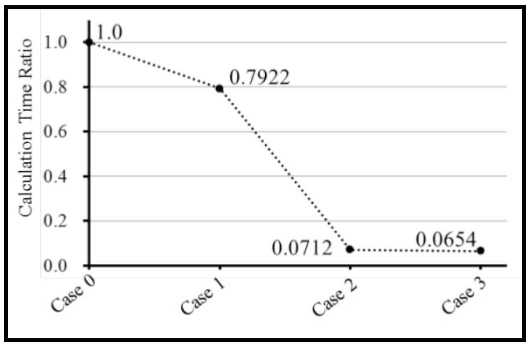


Fig. 7. Calculation Time Reduction Rate

5. Conclusion

In this paper, we proposed how to detect plagiarism of student reports that are becoming a social problem. First, we defined characteristic of text which we consider as plagiarism. And then we explained the plagiarism patterns often found in student reports.

The basic method for plagiarism detection is to generate one-dimensional string from two texts to be compared respectively, and repeat shifting and mutual comparison. This is a simple and steady method, however, it has disadvantages that the amount of calculation is large, and lack of robustness against small modification like word exchange to escape from plagiarism detection. So we developed some heuristics to overcome these disadvantages. These are consisted of early judgment of detection possibility, string substitution, and string compression. In this paper, we combined these heuristics with the above basic detection method and proposed it as a plagiarism detection method.

In addition, the proposed method simultaneously calculates the composition rate of the plagiarized section which named plagiarism rate. Therefore, it's possible to automatically eliminate worthless reports by previously determining a threshold to be rejected.

Finally, we showed that the developed heuristics can improve the accuracy of plagiarism detection and significantly reduce the calculation time. And also we explained this whole detection method is practical in our task.

The proposed method is designed to work well against assumed plagiarism patterns, but there is also a possibility that various patterns of plagiarism will emerge in the future. It's necessary to keep catching up with these unknown new patterns.

References

[1] A. Parker et al: (1989) “Computer algorithms for plagiarism detection”, IEEE Transactions on Education Vol.32 Issue 2: 94-99.

[2] M.K.Vijaymeena et al: (2016) “A SURVEY ON SIMILARITY MEASURES IN TEXT MINING “, Machine Learning and Applications: An International Journal Vol.3, No.1: 1412-1419.

[3] Salha M. Alzahrani et al: (2012) “Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods”, IEEE Transactions on Systems, Man, and Cybernetics, Part C Vol.42 Issue 2: 133 – 149.

[4] Yuuki Mori et al: (2015) “Developing a Plagiarism Detection System based on Citations for Academic Reports”, Information Processing Society of Japan, Kansai-Branch Convention, C-05 (in Japanese).
[5] Kayano Umezawa et al: (2011) “A Study on Identifying Similar Documents based on the Dimension Reduction of a Document Vector”, The 3rd Forum on Data Engineering and Information Management, A6-1 (in Japanese).
[6] Anna Huang: (2008) “Similarity measures for text document clustering”, Proceedings of the sixth New Zealand Computer Science Research Student Conference: 49-56.
[7] Alberto Barr´on-Cede˜no et al: (2009) “On Automatic Plagiarism Detection Based on n-Grams Comparison”, European Conference on Information Retrieval, Advances in Information Retrieval: 696-700.
[8] Marcin Kuta et al: (2014) “Optimisation of Character n-gram Profiles”, International Conference on Artificial Intelligence and Soft Computing, Artificial Intelligence and Soft Computing: 500-511.
[9] Efstathios Stamatatos: (2011) “Plagiarism detection using stopword n‐grams”, Journal of the American Society for Information Science and Technology 62(12): 2512-2527.
[10] Kazushi Ueta et al: (2010) "Plagiarism Detection methods based on Similarity by Distance for programing Reports", Information Processing Society of Japan, SIG Technical Report Vol.2010-CE-107 No.9 (in Japanese).
[11] Benno Stein et al: (2010) “Intrinsic plagiarism analysis”, Language Resources and Evaluation Vol.45 Issue 1: 63–82.