K-Gram As A Determinant Of Plagiarism Level In Rabin-Karp Algorithm

Andysah Putera Utama Siahaan, Mesran, Robbi Rahim, Dodi Siregar

Abstract: Rabin-Karp is one of the algorithms used to detect the similarity levels of two strings. In this case, the string can be either a short sentence or a document containing complex words. In this algorithm, the plagiarism level determination is based on the same hash value on both documents examined. Each word will form K-Gram of a certain length. The K-Gram will then be converted into a hash value. Each hash value in the source document will be compared to the hash value in the target document. The same number of hashes is the level of plagiarism created. The length of K-Gram is the determinant of the plagiarism level. By determining the proper length of K-Gram, it produces the accurate result. The results will vary for each K-Gram value.

Index Terms: Text Mining, Plagiarism, Similarity

1. Introduction

Rabin-Karp algorithm is an algorithm that is often used to determine whether a document is a plagiarism of other documents [1][2]. This algorithm works by determining the values of the existing snippets in the source and target documents. The more the same value on both documents, the higher the plagiarism level of the document [3][4]. But the plagiarism value of the two documents is not always the same. K-Gram is the determinant of the accuracy of a plagiarism analysis. The problem is how to determine this value correctly to approach a high accuracy value. The value of K-Gram will be different for each language used on the documents to be compared. This value is seen by looking at how short the number of characters used in each document. English will be more likely to match the small value of K-Gram compared to Bahasa Indonesia. K-Gram is often a dilemma to determine the truth of a document. Not always this value produces reality. The disadvantage of document comparisons is the inability of an application to determine which documents are correct and which documents first come to the surface. But this method can help an analyst to determine the level of similarity of some documents based on the value of K-Gram used. This study aims to compare the value of K-Gram used against several documents. From these calculations, it can be seen which K-Gram values are more suitable for a particular case.

- *Andysah Putera Utama Siahaan, Mesran, Robbi Rahim, Dodi Siregar*
- *Faculty of Computer Science, Universitas Pembanguan Panca Budi, Medan, Indonesia*
- *Department of Computer Engineering, STMIK Budi Darma, Medan, Indonesia*
- *Department of Health Information,, Akademi Perekam Medik dan Infokes Imelda, Medan, Indonesia*
- *Department of Informatics, Sekolah Tinggi Teknik Harapan, Medan, Indonesia*
- *Student of Universiti Malysia Perlis, Kangar, Malaysia*

2. Related Work

In 2014, Ashish Prosad Gope and Rabbi Narayan Behera conducted research on Rabin-Karp's relationship with DNA. Each DNA has its own sequence. This study is a field of bioinformatics. DNA sequence is a sequence of characters that have a certain length to determine the nucleotides present in the DNA. In this sequence, some information related to the disease in humans is obtained. This sequence has a certain pattern so that an algorithm is needed to match the pattern [6]. This study evaluated four performance pattern matching algorithms and then produced a new algorithm based on Rabin Karp's algorithm. This technique ensures that the comparison of characters can be eliminated from the Rabin-Karp algorithm. This algorithm looks for the patterns specified in a large set of DNA sequences [6].

2.1 Example of Improved Rabin-Karp

The following figure is an example of the development of the Rabin-Karp algorithm developed by both researchers.

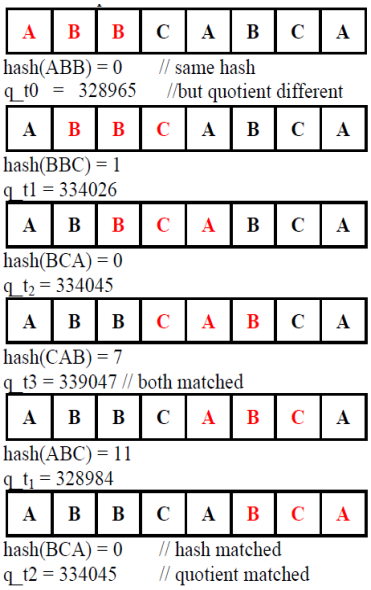


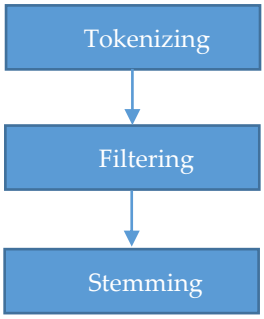
Fig. 1 *Improved Rabin-Karp Algorithm*

Figure 1 describes since the hash = 0 and quotient = 334045, both matched. Only "BCA" pattern is matched. And hash (ABB) = 0 and quotient = 328965, which has not matched, ABB is not compared. In some problems, this algorithm has no significant

difference with the pure Rabin-Karp algorithm. The complexity of this algorithm can be significantly improved [5]. The complexity of time during the worst case is O ((n-m + 1) m) to O (nm + 1). The complexity of this time depends on the prime number being the modulo. Choosing the right prime number will improve the optimization is good [6].

3. IMPLEMENTATION

Some steps must be taken before generating plagiarism value. Not all words will be used as input. These processes are Tokenizing, Filtering and Stemming. This should be done to streamline words and discard words that do not become keywords in comparison.



In this section, it will be examined two different sentences. The experiment will be done several times with different K-Gram values. The first sentence is "**plagiarism is an act or instance of using or closely imitating the language and thoughts of another author without authorization**". The second sentence is "**plagiarism is an act of copying the ideas or words of another person without giving credit to that person**".

Table 1 *Word Extraction*

First Sentence	Second Sentence
plagiarism	plagiarism
instance	copying
using	ideas
closely	words
imitating	another
language	person
thoughts	without
another	giving
author	credit
without	that
authorization	person

Table 1 illustrates the extraction of important words from the first and second sentences. Not all words will be taken from a sentence. Some processes must be skipped so as to produce important words only.

3.1 K-Gram = 10

Table 2 *K-Gram = 10*

1st Sentence	Hash	2st Sentence	Hash
plagiarism	7208	plagiarism	7208
lagiarismi	3721	lagiarismc	3715
agiarismin	8470	agiarismco	8411
giarismins	9795	giarismcop	9202
iarisminst	3653	iarismcopy	7735
arisminsta	2463	arismcopyi	3263
risminstan	9762	rismcopyin	7755
isminstanc	9448	ismcopying	9396
sminstance	375	smcopyingi	9866
minstanceu	759	mcopyingid	5589
instanceus	3976	copyingide	2227
nstanceusi	5701	opyingidea	7596
stanceusin	8463	pyingideas	2504
tanceusing	1569	yingideasw	6730
anceusingc	7781	ingideaswo	4882
nceusingcl	2898	ngideaswor	4763
ceusingclo	448	gideasword	9080
eusingclos	9838	ideaswords	6509
usingclose	3861	deaswordsa	1002
singclosel	5796	easwordsan	466
ingclosely	4938	aswordsano	214
ngcloselyi	5314	swordsanot	7292
gcloselyim	4592	wordsanoth	9874
closelyimi	1647	ordsanothe	6084
loselyimit	1815	rdsanother	7397
oselyimita	9411	dsanotherp	5825
selyimitat	641	sanotherpe	8659
elyimitati	3407	anotherper	3540
lyimitatin	9609	notherpers	523
yimitating	7301	otherperso	6719
imitatingl	582	therperson	3736
mitatingla	1774	herpersonw	9457
itatinglan	4114	erpersonwi	5369
tatinglang	7079	rpersonwit	9221
atinglangu	2857	personwith	4043
tinglangua	3682	ersonwitho	2098
inglanguag	8901	rsonwithou	6533
nglanguage	4912	sonwithout	7196
glanguaget	579	onwithoutg	8913
languageth	1544	nwithoutgi	5657
anguagetho	6715	withoutgiv	8031
nguagethou	2254	ithoutgivi	7672
guagethoug	4007	thoutgivin	2638
uagethough	5803	houtgiving	8468
agethought	5210	outgivingc	5480
gethoughts	7216	utgivingcr	1357
ethoughtsa	7865	tgivingcre	763
thoughtsan	4154	givingcred	9729
houghtsano	3622	ivingcredi	2982
oughtsanot	7072	vingcredit	5779
ughtsanoth	7260	ingcreditt	70
ghtsanothe	9758	ngcreditth	6668
htsanother	3281	gcredittha	8113
tsanothera	3643	creditthat	6847
sanotherau	8525	reditthatp	3776
anotheraut	2202	editthatpe	9632

notherauth	7146	ditthatper	1814
otherautho	2900	itthatpers	8591
therauthor	5578	tthatperso	1822
herauthorw	7863	thatperson	315
erauthorwi	9443		
rauthorwit	9933		
authorwith	1156		
uthorwitho	6700		
thorwithou	4174		
horwithout	3827		
orwithouta	9103		
rwithoutau	7569		
withoutaut	7549		
ithoutauth	2851		
thoutautho	4464		
houtauthor	6725		
outauthori	8070		
utauthoriz	7251		
tauthoriza	9664		
authorizat	8692		
uthorizati	1998		
thorizatio	7183		
horization	3890		

The table above shows the hash value of K-Gram = 10. From the hash values generated by each sentence, the same values will be obtained. There are 79 hash values in the first sentence and there are 60 hash values in the second sentence. So the total hash is 139. There is only one has same value in both tables. The value is 7208. This calculation can be seen below:

Mod = 10007
Base = 10
Hash = [112*10^9] + [108*10^8] + [97*10^7] + [103*10^6]
+ [105*10^5] + [97*10^4] + [114*10^3]
+ [105*10^2] + [115*10^1] + [109*10^0]
Hash = 123884595759 Mod 10007
Hash = 7208

The calculation of the plagiarism level does not end at the hash value only. There is a formula for determining the percentage of similarity of the two comparable documents. The formula used is as follows:

$$P = \frac{2 * SH}{THA + THB} * 100\%$$

Where:
P = Plagiarism Rate
SH = Identical Hash
THA = Total Hash in Document A
THB = Total Hash in Document B

In the previous calculation there is one hash that has the same value. So the plagiarism level calculation is as follows.

P = $\frac{2*1}{79+60} * 100\%$
= $\frac{2}{139} * 100\%$
= 0,0143884892086331 * 100%
= 1,439%

3.2 K-Gram = 5

Table 2 *K-Gram = 5*

1st Sentence	Hash	2st Sentence	Hash
plagi	7974	plagi	7974
lagia	7621	lagia	7621
agiar	3828	agiar	3828
giari	5147	giari	5147
iaris	8760	iaris	8760
arism	4996	arism	4996
rismi	6820	rismc	6814
ismin	6241	ismco	6182
smins	9833	smcop	9240
minst	6426	mcopy	501
insta	1938	copyi	2738
nstan	6826	opyin	4399
stanc	6010	pying	1828
tance	8209	yingi	6211
anceu	264	ingid	631
nceus	9545	ngide	3754
ceusi	3185	gidea	5309
eusin	8869	ideas	373
using	5800	deasw	1192
singc	6247	easwo	9024
ingcl	579	aswor	7361
ngclo	3244	sword	444
gclos	227	words	2598
close	9581	ordsa	4386
losel	2778	rdsan	1705
osely	5440	dsano	5127
selyi	2233	sanot	8351
elyim	468	anoth	1601
lyimi	1848	nothe	2887
yimit	6142	other	6663
imita	9945	therp	4463
mitat	6846	herpe	2816
itati	6146	erper	5533
tatin	8878	rpers	2473
ating	6940	perso	2800
tingl	6249	erson	5929
ingla	658	rsonw	6437
Nglan	4033	sonwi	2406
Glang	8105	onwit	2205
langu	8321	nwith	9910
angua	804	witho	6841
nguag	4926	ithou	6808
guage	7026	thout	5497
uaget	7537	houtg	3151
ageth	3608	outgi	8874
getho	2953	utgiv	6565
ethou	6836	tgivi	3896
thoug	5484	givin	7162
hough	3022	iving	8884
ought	7595	vingc	6226
ughts	3779	ingcr	585
ghtsa	6049	ngcre	3294
htsan	7768	gcred	712
tsano	5015	credi	4428
sanot	8351	redit	1291
anoth	1601	editt	992

nothe	2887	ditth	7087
other	6663	ittha	7918
thera	4448	tthat	6590
herau	2682	thatp	4083
eraut	4195	hatpe	9023
rauth	9096	atper	7561
autho	8988	tpers	2459
uthor	6721	perso	2800
thorw	5470	erson	5929
horwi	2883		
orwit	6205		
rwith	9882		
witho	6841		
ithou	6808		
thout	5497		
houta	3145		
outau	8826		
utaut	6083		
tauth	9082		
autho	8988		
uthor	6721		
thori	5456		
horiz	2760		
oriza	4956		
rizat	7411		
izati	2139		
zatio	8837		
ation	6957		

The table above shows the hash value of K-Gram = 5. There are 84 hash values in the first sentence and there are 65 hash values in the second sentence. So the total hash is 149. There are 13 values have the same value in both tables. The plagiarism calculation can be seen below:

$$P = \frac{2*13}{84+65} * 100\%$$
$$= \frac{26}{149} * 100\%$$
$$= 0{,}174496644295302 * 100\%$$
$$= 17{,}449\%$$

4. CONCLUSION

The determination of the value of K-Gram greatly affects the percentage of truth plagiarism level. Every language in the world has different K-Gram values. For higher K-Gram values tend to have a low level of similarity, while lower K-Gram values increase the percentage of similarity. But this is not absolute as a determinant of the document's authenticity. This method can help analysts to see more about the techniques used to compare documents. Determination of K-Gram requires correct consideration so that the value will be more accurate. This method is only used to help experts in analyzing. All decisions will return to the specialist.

REFERENCES

[1]. S. K. Shivaji and P. S., "Plagiarism Detection by using Karp-Rabin and String Matching Algorithm Together," International Journal of Computer Applications, vol. 116, no. 23, pp. 37-41, 2015.

[2]. M. Cebrián, M. Alfonseca and A. Ortega, "Towards the Validation of Plagiarism Detection Tools by Means of Grammar Evolution," IEEE Transactions on Evolutionary Computation, vol. 13, no. 3, pp. 71-77, 2009.

[3]. A. Parker and J. O. Hamblen, "Computer Algorithm for Plagiarism Detection," IEEE Trans. Education, vol. 32, no. 2, pp. 94-99, 1989.

[4]. A. Apostolico, String editing and Longest Common Subsequences, vol. 3, Germany: Springer-Verlag, 1997, pp. 1-10.

[5]. Sunita, R. Malik and M. Gulia, "Rabin-Karp Algorithm with Hashing a String Matching Tool," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 3, pp. 389-392, 2014.

[6]. A. P. Gope and R. N. Behera, "A Novel Pattern Matching Algorithm in Genome," International Journal of Computer Science and Information Technologies, vol. 5, no. 4, pp. 5450-5457, 2014.