Implementation of Rabin Karp Algorithm in E-Commerce Search Box Feature (Case Study: Sinar Baja Store)

[1*]Yazeed Qholili Arifin, [2]Ade Ismail, [3]Vipkas Al Hadid Firdaus
Politeknik Negeri Malang, Indonesia
[1]yazeedarifin@gmail.com, [2]ismail@polinema.ac.id, [3]vipkas@polinema.ac.id
***Corresponding Author**

Abstract— Inside any e-commerce platform, search features are a key factor in an online business. In order to look for the desired item name, users need to type a pattern into the search feature. Inside the typing process users may make some mistypes. Based on the problem, the writer tried to create a search box as the search feature inside an e-commerce system with a capability to look for item names, even if there are some accidental mistypes inside the user's typing process. The main purpose is to reduce the possibility of empty results after a searching process due to the user's accidental mistypes. This research utilizes Rabin Karp Algorithm in order to look for the item names. The result of this research is that the Rabin Karp Algorithm can be implemented in a website-based e-commerce, but this method is slower than SQL as a method to search item names inside the same system. It happens because Rabin Karp requires a longer process than SQL to get the similarity percentage from each comparison. This research shows that k-gram value inside the algorithm affects the overall results with a condition where greater the value.

Keywords— e-commerce, Rabin Karp algorithm, search box, search feature

I. INTRODUCTION

E-commerce is a shape of business relationship where it has interaction between the existing actors by utilizing the internet technologies [1]. Since there are many items in the e-commerce platform, people need to scroll their screen from top to bottom. Because of it, people will need a better solution to search for their desired item sooner. A search feature will be a good solution for that problem because search feature is a key factor in an online business [2]. Real-word errors might happen when a user mistakenly inserts a word which is correctly spelled, while intending another word [3]. People made 2.29 error corrections for each sentence, and with some amount of people pressing Backspace or Delete up to 8.5 times for each sentence [4].

In the previous research made by Herriyance, Handrizal, and Siti Dara Fadilla a string-matching algorithm namely Rabin Karp Algorithm is successfully implemented into a general dictionary application with average running time 14.9ms for 10 times word searching trial. But the base number (K-Gram) can affect the running time process. Bigger the number, longer the running time process [5]. The next research was made by Andysah Putera Utama Siahaan where the Rabin Karp Algorithm can check an image similarity based on hash value calculation where the calculation itself works on the same way for string matching [6]. The next is Asvarizal Filcha and Mardhiya Hayaty said in their research that Rabin Karp Algorithm gives a better time efficiency in order to look for strings with multiple patterns with 90% accuracy calculation on document plagiarism detector of students work which comes from 20 comparison of students work documents [7]. Then, research made by Muhamad Arief Yulianto and Nurhasanah proves the accuracy of the Rabin Karp algorithm on the word similarity test is increased by 20.06% through the implementation of the Jaro-Winkler on Rabin Karp [8]. Another research said that string length in a test affects the execution time to generate similarity percentage [9].

ISSN 2442-7888 (online) DOI 10.24167/Sisforma

Based on previous research, the author is going to utilize the Rabin Karp Algorithm in order to search item names. This algorithm is good to get text patterns because it uses hash values as parameter to do a comparison. The data of this research is going to be metal-related item names. The items will be arranged based on the similarity value from the top to the bottom. As an example, when somebody typed 'aluminium' but the desired word is 'alumunium'. Right after submission of the word, that person will get nothing from the search feature. It happens because the search box feature looks for only the same exact word inside the database. In order to get the desired items even after mistype applied, a suggestion of a mistyped keyword is necessary. Users may wonder about the search feature since it doesn't return any answer without realizing their mistype.

II. METHOD

A. Preprocessing

The preprocessing stage of this research is done to transform strings into lowercase form, and to remove unnecessary punctuation. This step starts before Rabin Karp Algorithm to prepare data. The purpose of preprocessing is to prepare texts into a processible data within the next steps [10].

B. Rabin Karp Algorithm

Rabin Karp Algorithm is a kind of string-matching algorithm which uses hashing inside a string-matching process to find pattern inside a text [7]. If there are 2 hash values and they match, both will get one more comparison for each character in the string text.

C. K-Gram

K-gram is to make transform a string pattern to become sub-patterns with a specific length by using a determined value [11]. In this case, the k-gram value also becomes minimum string length for each comparison.
Example:
Words : kursinyaadi
K-Gram : 4
Result : {kurs} {ursi} {rsin} {siny} {inya} {nyaa} {yaad} {aadi}

D. Hashing

Hashing is a function convert sub-strings into numbers where the output will be called as hash [12]. It actually works by using the formula below.

$$H(_{c1\ldots\ldots ck}) = c1 * b^{(k-1)} + c2 * b^{(k-2)} + \cdots + c_{(k-1)} * b^{k} + c^{k} \quad \ldots (1)$$

Explanation:
H: substring
c: ASCII number per-character
b: prime number constant
k: amount of character
q: prime number modulo

E. Dice Similarity

Dice Similarity is a step to get similarity value of two different strings[13]. The formula of this similarity is written below:

$$S = 2C/(A+B) \quad \ldots (2)$$

Explanation:
S: Similarity
A: Amount of hashed K-Gram from string number 1
B: Amount of hashed K-Gram from string number 2
C: Amount of K-Gram with the same exact hash number from both strings

F. Flowchart

Flowchart is a figure to identify steps inside a process and also to find out any potential of weak spots inside the process [14]. According to the attached reference, in order to identify the algorithm process before a development, the author made several flowcharts which are the flowchart of preprocessing, Rabin Karp Algorithm, and Dice Similarity. The flowcharts will be listed respectively as follows.

ISSN 2442-7888 (online) DOI 10.24167/Sisforma

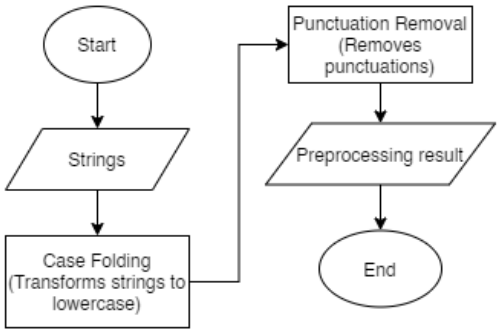

Figure 1. Preprocessing Flowchart

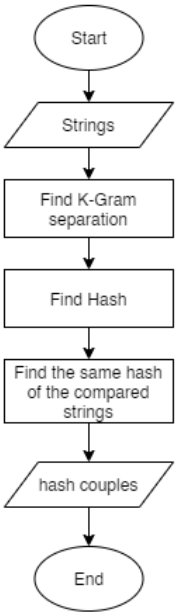

Figure 2. Rabin Karp Algorithm Flowchart

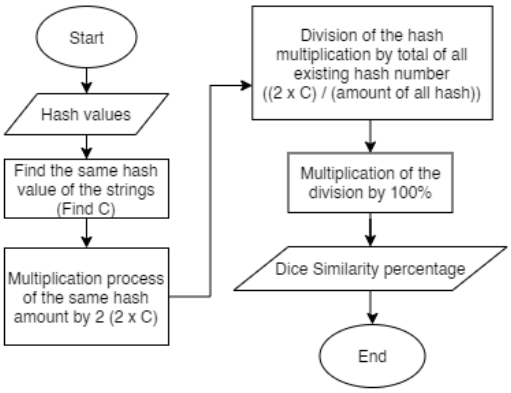

Figure 3. Dice Similarity Flowchart

G. Software Development Methodology

This research uses the Waterfall Model. Waterfall is a classical model that is systematic, sequential in building software [15].

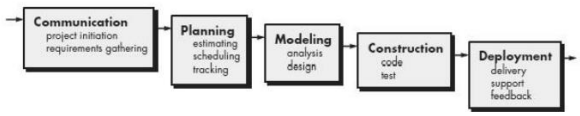

Figure 4. Waterfall Development Model

The steps of the sequence above can be described as follows:

1. Communication
   This step is used to obtain the system requirements from a meeting with customers or reviewing journals and articles from the internet for more references.
2. Planning
   Planning process to generate user requirement document or any details related to user needs in software making based on the communication process or the previous step.
3. Modelling
   The modelling process is used to translate the gathered requirement into a predictable software design before coding.
4. Construction
   Construction can be translated as coding process. Coding is a translation of design in a language that can be recognized by a computer. After coding there will be testing for the developed system which is purposed to find errors for later repair.
5. Deployment
   Deployment is the final stage in software development. In this stage the system will be used by the user and periodic maintenance of the software will be performed.

III. RESULTS AND DISCUSSION

Below is containing both result and discussion of this research.

A. RESULT

The implemented data inside Rabin Karp Algorithm is item names. Before the algorithm starts, the data will enter preprocessing stage to remove any

ISSN 2442-7888 (online) DOI 10.24167/Sisforma

unnecessary characters and set the data into lowercase mode. Right after the preprocessing stage is finished, Rabin Karp Algorithm will take place to get the fingerprints or hash couples. Dice Similarity Method will transform the fingerprints into similarity percentage to the item names respectively. Figure 4 represents the process output and Figure 5 represents both of similarity percentage and the rest of item detail data which can be seen at inspect element feature of a browser.

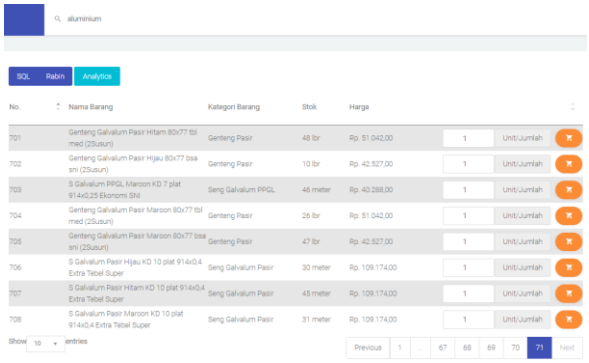

Figure 5. Rabin Karp Algorithm Processing Result

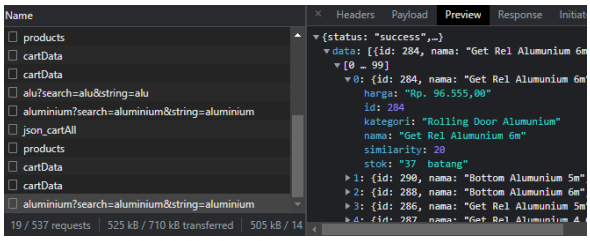

Figure 6. Similarity Percentage and Item Data Response

The platform to implement Rabin Karp Algorithm was developed by using PHP as the server-side programming language, Laravel as the framework, JavaScript as the client-side programming language, jQuery as the interface library, and MariaDB as the database.

B. DISCUSSION

It takes 10 times of system searching process to get the average estimation time within the same circumstances. As a user input, writer will use the same exact keyword which exist within the data inside database for both SQL and Rabin Karp Algorithm as methods to get item names among 3359 item names inside the system.

Table 1. Time Response Table

No.	SQL Duration	Rabin Karp Duration
1	0.22s	1.37s
2	0.23s	0.93s
3	0.24s	1.01s
4	0.24s	0.96s
5	0.25s	1.13s
6	0.23s	0.93s
7	0.24s	0.91s
8	0.24s	0.90s
9	0.23s	0.92s
10	0.25s	1.00s

Based on the obtained data inside Table 1, the average elapsed time for SQL is 0.24s and Rabin Karp is 1.01s. Rabin Karp takes longer time than SQL query to get the result because inside the system, the algorithm requires SQL statement to obtain data to compare from the database and several steps of the algorithm itself to do in order to get similarity percentage between the data from database and input from user. The pure SQL itself can get the item name directly with a few lines of SQL script but it doesn't get any similarity percentage from the process, unlike Rabin Karp. Similarity percentage is used to define the similarity level between 2 compared data.

Inside the database there are many items with "alumunium" as a part of their names and customers can simply type that particular word as a keyword. But this test case will use "aluminium" instead of "alumunium" in order to know whether the searching result is affected by the k-gram or not. If only there is a single sub-pattern or even more exist within the item name inside the database, the system will display it into customer searching result. K-gram values are 3, 4, and 5 will be used sequentially to get the comparison in ascending order by k-gram. Below is the list of k-gram test with their respective value:

a. K-Gram 3

ISSN 2442-7888 (online) DOI 10.24167/Sisforma

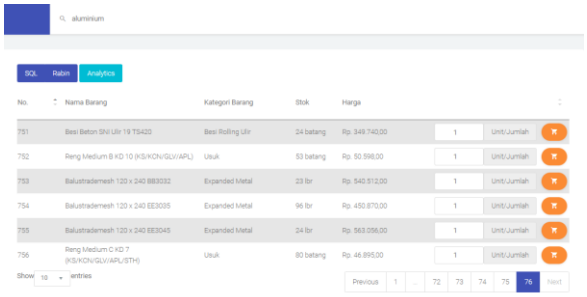

Figure 7. K-Gram 3 Interface Result

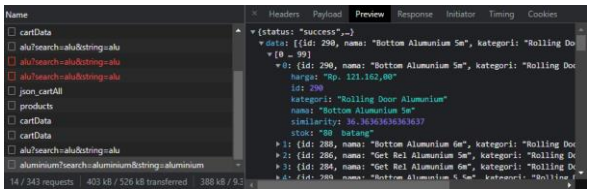

Figure 8. K-Gram 3 Highest Similarity Percentage Response

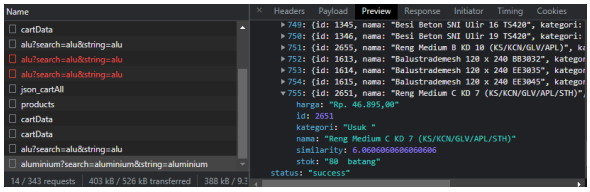

Figure 9. K-Gram 3 Lowest Similarity Percentage Response

Number of item names : 756
Highest. similarity : 36.3637 %
Lowest. similarity : 6.0607 %

b. K-Gram 4

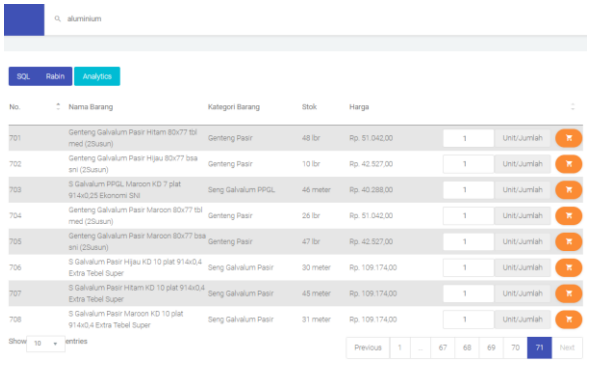

Figure 10. K-Gram 4 Interface Result

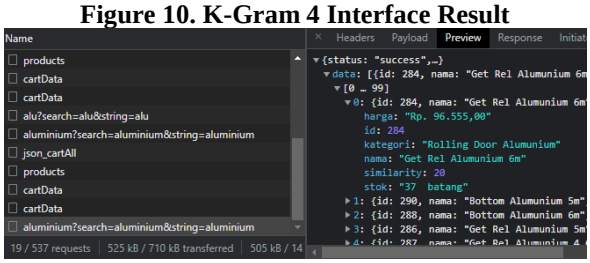

Figure 11. K-Gram 4 Highest Similarity Percentage Response

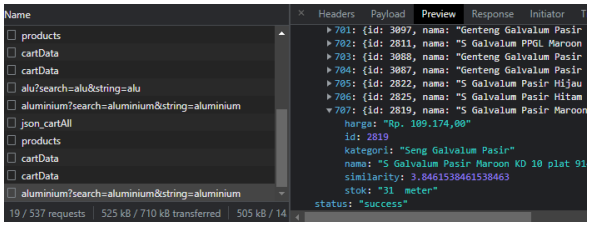

Figure 12. K-Gram 4 Lowest Similarity Percentage Response

Number of item names : 708
Highest. similarity : 20 %
Lowest. similarity : 3.8462 %

c. K-Gram 5

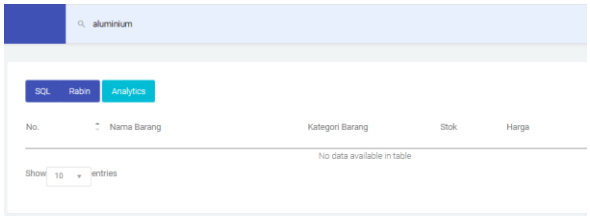

Figure 13. K-Gram 5 Interface Result

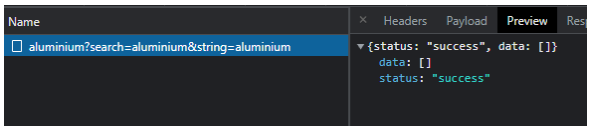

Figure 14. K-Gram 5 Response

Number of item names : 0
Highest. similarity : _
Lowest. similarity : _

Based on the data above, k-gram value is a vital point to determine the algorithm sensitivity in data processing which affects both number of item names and similarity percentage. The greater the value, less sensitive it becomes.

User Acceptance Testing (UAT) is used to assess the user's understanding about the application which is made to answer a problem [16]. The test form was given to the public, so people can give a try to the developed search feature within the e-commerce app when they are trying to look for metal related item names. An excel-file link of the metal item names is attached within the question form to inform the users about the existing item names within the system to search. The test was given to 11 respondents with customer role because the developed feature was meant for customer to look for item names. Based on the UAT result, the customers were fully helped by the

ISSN 2442-7888 (online) DOI 10.24167/Sisforma

suggestion provided by the system to avoid typing mistakes. At the same time, it proves that the algorithm can be developed within the existing software.

IV. CONCLUSION

Rabin Karp Algorithm can be implemented to find hashes among the inserted string inside the search box and strings inside the database, but it takes more time to finish compared to common SQL script to find data with the same pattern. K-gram value within Rabin Karp Algorithm affects the overall searching result with a condition where greater the value, fewer item names and similarity percentage can be obtained. It can be implemented into a web-based e-commerce by using PHP as the server-side programming language, Laravel as the framework, JavaScript as the client-side programming language, jQuery as the interface library, and MariaDB as the database.

REFERENCES

[1] V. Babenko, Z. Kulczyk, I. Perevosova, O. Syniavska, and O. Davydova, "Factors of the development of international e-commerce under the conditions of globalization," *SHS Web Conf.*, vol. 65, p. 04016, 2019.

[2] A. Maros, F. Belém, R. Silva, S. Canuto, J. M. Almeida, and M. A. Gonçalves, "Image aesthetics and its effects on product clicks in e-commerce search," *CEUR Workshop Proc.*, vol. 2410, 2019.

[3] S. M. S. Dashti, "Real-word error correction with trigrams: correcting multiple errors in a sentence," *Lang. Resour. Eval.*, vol. 52, no. 2, pp. 485–502, 2018.

[4] V. Dhakal, A. M. Feit, P. O. Kristensson, and A. Oulasvirta, "Eystrokes," *Proc. 2018 CHI Conf. Hum. Factors Comput. Syst. - CHI '18*, pp. 1–12, 2018.

[5] A. Rabin-Karp Pada Kamus Umum Berbasis Android and S. Dara Fadilla, "Analisis Algoritma Rabin-Karp Pada Kamus Umum Berbasis Android," *J. Ris. Sist. Inf. Dan Tek. Inform.*, no. 2, 2017.

[6] A. P. U. Siahaan, "Rabin-Karp Elaboration in Comparing Pattern Based on Hash Data," *Int. J. Secur. Its Appl.*, vol. 12, no. 2, pp. 59–66, Mar. 2018.

[7] A. Filcha and M. Hayaty, "Implementasi Algoritma Rabin-Karp untuk Pendeteksi Plagiarisme pada Dokumen Tugas Mahasiswa (Rabin-Karp Algorithm Implementation to Detect Plagiarism on Student's Assignment Document)."

[8] M. A. Yulianto and N. Nurhasanah, "The Hybrid of Jaro-Winkler and Rabin-Karp Algorithm in Detecting Indonesian Text Similarity," *J. Online Inform.*, vol. 6, no. 1, p. 88, Jun. 2021.

[9] R. Hidayat, H. Haryanto, and Y. A. Pramono, "Design of Assesment Information System Employee Service in PT. Aerotrans Services Indonesia with Methods Key Performance Indicator (KPI)," *REMIK (Riset dan E-Jurnal Manaj. Inform. Komputer)*, vol. 4, no. 1, p. 5, Sep. 2019.

[10] Sugiono, Herwin, Hamdani, and Erlin, "View of Aplikasi Pendeteksi Tingkat Kes...ks_ Algoritma Rabin Karp Vs. Winnowing."

[11] Maskur and Deny Qutara Putra, "Deteksi Kemiripan Dokumen Proposal Penelitian dan Pengabdian Menggunakan Algoritma Biword Winnowing."

[12] A. Bahrul Khoir, H. Qodim, B. Busro, and A. Rialdy Atmadja, "Implementation of rabin-karp algorithm to determine the similarity of synoptic gospels," *J. Phys. Conf. Ser.*, vol. 1175, no. 1, 2019.

[13] S. Bahri and R. Wajhillah, "Optimalisasi Algoritma Rabin Karp menggunakan TF-IDF Dalam Pencocokan Text Pada Penilaian Ujian Essay Otomatis," *Jl. Cemerlang*, vol. 4, no. 2, 2020.

[14] M. Borgert, J. Binnekade, F. Paulus,

ISSN 2442-7888 (online) DOI 10.24167/Sisforma

A. Goossens, and D. Dongelmans, "A flowchart for building evidence-based care bundles in intensive care: Based on a systematic review," *Int. J. Qual. Heal. Care*, vol. 29, no. 2, pp. 163–175, 2017.

[15] U. Ependi, N. Oktaviani, J. Jenderal Ahmad Yani No, and P. Palembang, "Abstract Keyword Searching with Knuth Morris Pratt Algorithm," *Sci. J. Informatics*, vol. 4, no. 2, pp. 2407–7658, 2017.

[16] I. P. A. E. Pratama, "The Implementation and Testing of Online Self-Diagnose Covid19 Application Using CBR and UAT," *Int. J. Adv. Data Inf. Syst.*, vol. 2, no. 2, pp. 73–83, 2021.

ISSN 2442-7888 (online) DOI 10.24167/Sisforma