LONTAR KOMPUTER VOL. 10, NO. 3 DECEMBER 2019 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2019.v10.i03.p07 **e-ISSN 2541-5832**
Accredited B by RISTEKDIKTI Decree No. 51/E/KPT/2017

Query Suggestion on Drugs e-Dictionary Using the Levenshtein Distance Algorithm

Halimah Tus Sadiah[a1], Muhamad Saad Nurul Ishlah[a2], Nisa Najwa Rokhmah[b3]

[a]Manajemen Informatika, Universitas Pakuan
Jl.Pakuan, Bogor 16143
[1]sadiahht@unpak.ac.id (Corresponding author)
[2]nurul.ishlah@unpak.a.c.id

[b]Farmasi, Universitas Pakuan
Jl.Pakuan, Bogor 16143
[3]nisanajwarokhmah@gmail.com

Abstract

The dictionary of medicine in the form of a thick book has many disadvantages, one of which is impractical. This is the reason for Indonesian developers to create a drugs e-Dictionary. But the drug e-Dictionary that has been developed is still in the form of a letter index so that users must search the terms one by one in sequential order. This has become so inefficient and ineffective that it is necessary to add a search function and query suggestion feature to the drug e-dictionary. The purpose of this study is to build a query suggestion facility on drugs e-Dictionary using the Levenshtein Distance algorithm. The stages of this research consist of the Development of web-based drugs e-Dictionary, Implementation of the Levenshtein Distance Algorithm, Query Suggestion Testing, and Usage. The query suggestion function works by producing the closest word output contained in the database. Based on the results of the implementation of the Levenshtein Distance algorithm and test results, Drugs e-Dictionary can evaluate words that are not in the database. It reaches 90% accuracy of the inputted query, with 90% precision and 90% recall in the confusion matrix.

Keywords: *Query Suggestion, Drugs e-Dictionary, Algorithm, Levenshtein Distance Algorithm*

1. Introduction

The decision to use a drug (medication drug) always raise concern on the benefits and risks so that a Pharmacist needs a drug dictionary to search for previously unknown terms of medicine [1]. Besides, the drug dictionary becomes one of the learning tools that are used by Pharmacists, Students, and the Indonesian community in learning medicine or foreign terms about medicine. The drug dictionary that is used nowadays is in the form of a thick physical dictionary book. It turns out to have drawbacks, such as it is too heavy to be carried so that it is not practically handy. This is one of many reasons for Indonesian developers to compete in creating an electronic dictionary of drugs or what we know as the term drug e-dictionary.

Most of the available drugs e-dictionaries that have been developed so far are still in the form of a letter-index based dictionary. It makes users have to search for words or terms one by one in a sequential fashion. This has become so inefficient and ineffective that it is necessary to add a search function to the drug e-dictionary. The search function on drugs e-dictionary is very important because it can be used as a shortcut when searching words or terms needed so that users can search for words effectively and efficiently [2][3].

The drug e-dictionary search function needs to be optimized with the addition of the Query Suggestion facility. Query Suggestion is some interface between a user and a search engine [4]. This facility is an effective and efficient approach to help the user in the process of finding information by providing a suggestion for the user when mistyping is happened in the search form [2][3][5][6]. This feature is very important to be applied since it can improve the usability factor of searching [7][8]. It works by looking for the similarity between a correct query and a

false query in a database [9]. This feature can be a solution for preventing the user from typing the wrong name of the drug. The Query Suggestion can be used in a search application by implementing the Levenshtein Distance Algorithm.

Research on query suggestion has been done by Jiang et al. (2008), namely Query suggestion by query search: a new approach to user support in web search [3]. Meanwhile, research on the Levenshtein Distance algorithm was conducted by Ngafidin and Wibawanto (2015), namely the Implementation of the Autocomplete Feature and the Levenshtein Distance Algorithm to Increase the Effectiveness of Word Search in the Indonesian Big Dictionary (KBBI) [10]. This study aims to build a query suggestion facility using the Levenshtein Distance algorithm on drugs e-dictionary. This research is critical to do so that pharmacists, students, and the public can easily search for drug terms in the drug e-dictionary.

2. Research Methods

The research method used in this study consists of several stages, as shown in Figure 1, which is described below:

1. Development of Web-based e-Dictionary Drugs
   Development of Web-based e-Dictionary Drugs using the SDLC (System Development Life Cycle) method that has been adapted to the needs of Web-based Drugs e Dictionary [11][12]. The stages are Plan, Analysis, Design, Code, Testing.
2. Implementation of the Levenshtein Distance Algorithm
   The implementation is done by adding the Levenshtein Distance algorithm in the PHP programming language.
3. Testing Query Suggestion
   Testing is done by inputting drug terms in the search form as many as 100 terms. The number of terms entered consists of 50 correct terms, 50 incorrect terms, or incorrect terms.
4. Usage
   Drugs e-Dictionary that has been tested is then hosted to be used by users.

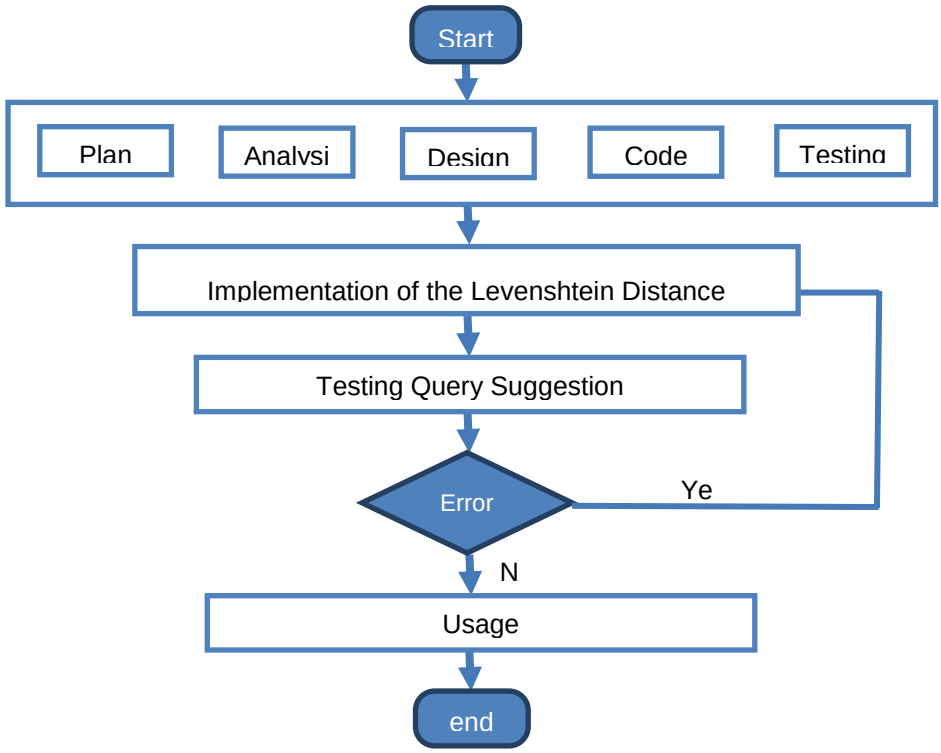

Figure 1. Research Method

3. Result and Discussion

3.1. Web-based Drugs e-Dictionary Development

3.1.1. Plan

In the planning stage, data collection is carried out. Data was collected from the ISO Indonesian Information Specialist book [13]. The collected data consist of drug categories, drug names, indications, contradictions, side effects, drug interactions, dosages, packaging, and drugs warning.

3.1.2. Analysis

In this stage, system functionality requirements and non-system requirements are collected. There are 28 system functionality requirements, namely 10 front end system functionality requirements and 18 back end system functionality requirements. The non-functional requirements only produced 7 system non-functional requirements.

3.1.3. Design

Next, in this design stage, a search system flow will be developed. It is depicted in Figure 2.

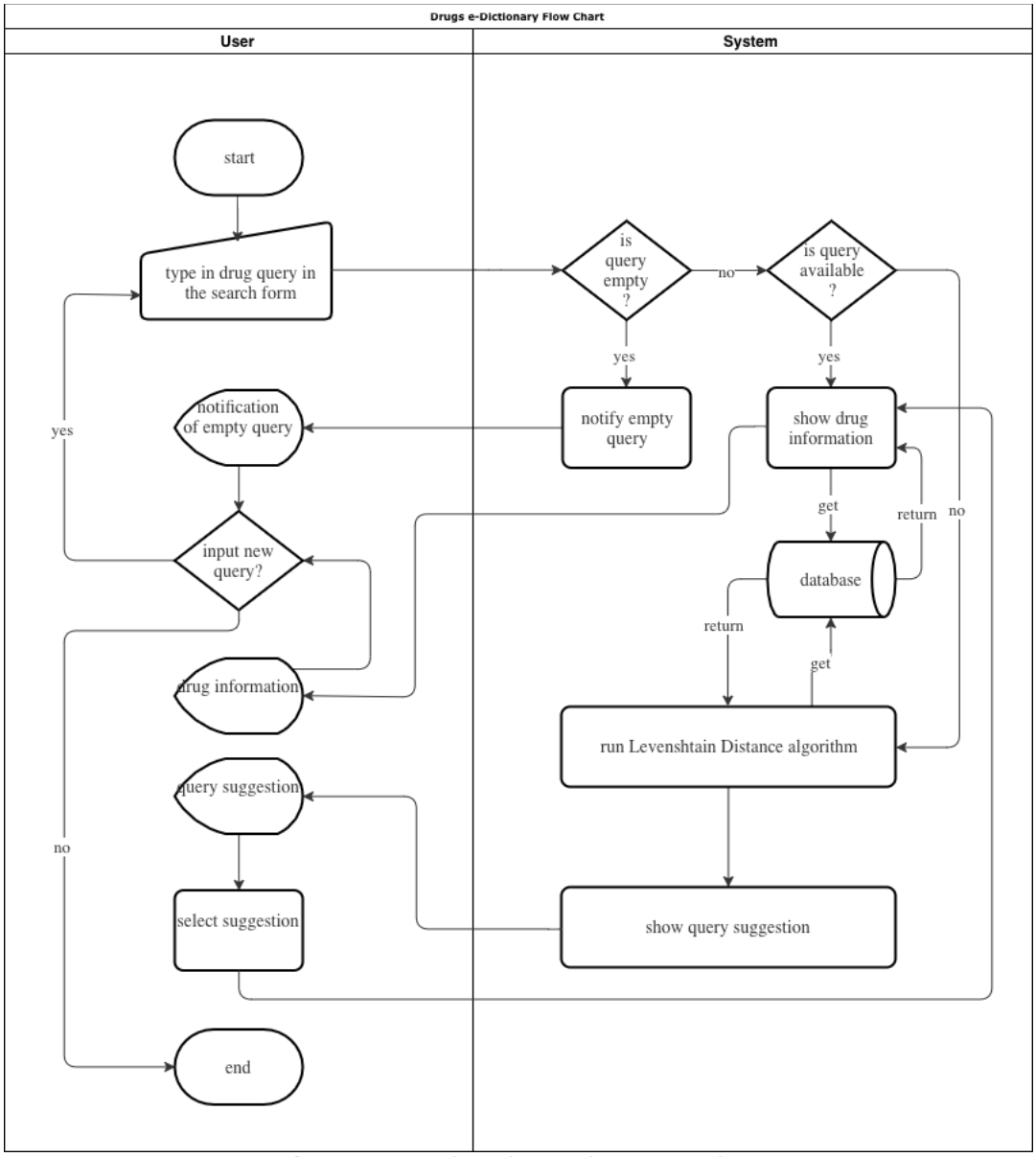

Figure 2. Developed Search System Flow

Based on Figure 2, the flow of drug searching is described below:

1. User accesses drugs e-dictionary website
2. User searches for the name or drug term in the search form
   - If the user's query is empty, then the system will show empty query notification or "query has not been inserted".
   - If the inputted query is in the database, then the system will show search results.
   - If the inputted query is not available in the database, then the system will proceed with Levenshtein Distance Algorithm followed up with the query suggestion

3.1.4. Code

In this implementation stage, the system is developed in PHP language with MYSQLi for the database connection. The result is a web *drug e-dictionary. Drugs e-dictionary* consists of the main searching page, which searches based on drugs term as depicted in figure 3; searching based on disease indication, as shown in Figure 4; and A-Z index-based searching, as depicted in Figure 5.

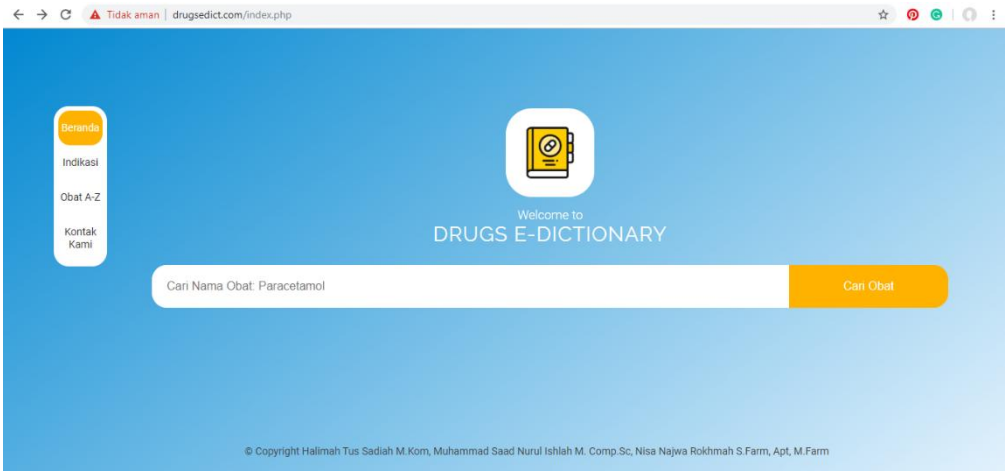

Figure 3. Drugs e-Dictionary website

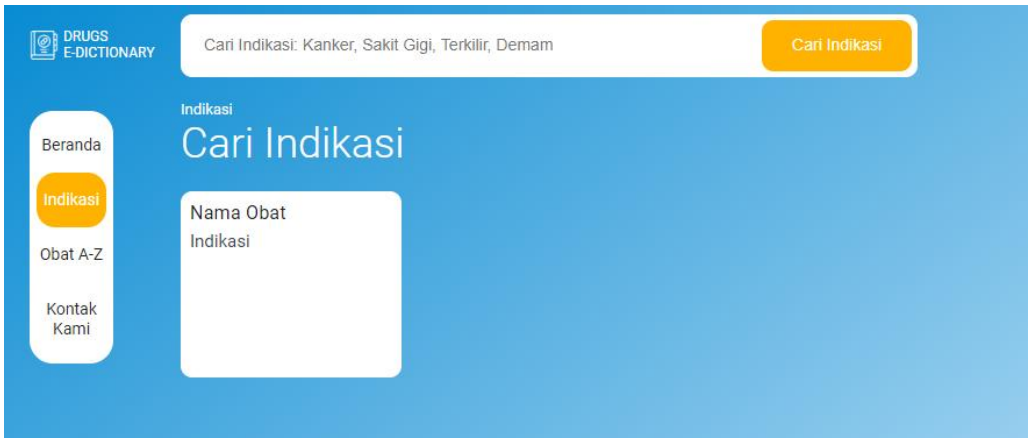

Figure 4. Homepage user interface based disease indication

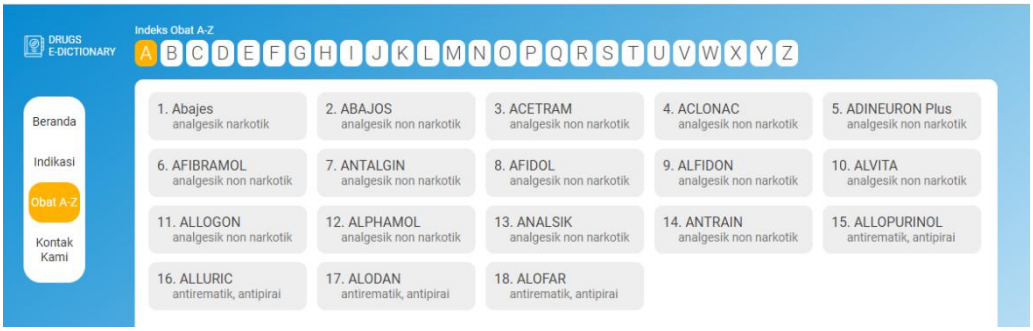

Figure 5. A-Z index-based searching page

3.1.5. Testing

A black box is used in the testing stage. It is usually called a system functional test [11]. Based on the test, 28 functions from the system is running as expected.

3.2. Levenshtein Distance Algorithm Implementation

The Levenshtein Distance algorithm is an algorithm created by Vladimir Levenshtein in 1965 [14]. This algorithm looks for the distance between the words entered by the user and the words stored in the system database by the method of calculating the number of differences between the two strings in the form of a matrix [15][16]. It works by calculating the distance between the two strings and then look for the minimum number of change operations to change from string A to string B. The calculation is represented using the Levenshtein distance calculation table, where the last value in the lower right corner is the final value of the second distance string. In the Levenshtein distance algorithm there are three operations performed, namely the operation of changing characters, adding characters and deleting characters [17][18] [19]. Figure 6 is a pseudocode Levenshtein distance algorithm.

```
1  int LevenshteinDistance(char s[1..m], char t[1..n])
2  {
3  // d is a table with m+1 rows and n+1 columns
4  declare int d[0..m, 0..n]
5  declare int cost
6  for i from 0 to m
7  d[i, 0] := i // deletion
8  for j from 0 to n
9  d[0, j] := j // insertion
10 for j from 1 to n
11 {
12 for i from 1 to m
13 {
14 if s[i] ≠ t[j] then
15 cost := 1
16 else
17 cost := 0
18 d[i, j] := minimum
19 (
20 d[i-1, j] + 1, // deletion
21 d[i, j-1] + 1, // insertion
22 d[i-1, j-1] + cost // substitution
23 )
24 }
25 }
26 return d[m,n]
```

Figure 6. Pseudocode Levenshtein Distance Algorithm

The pseudocode of the algorithm, as depicted in figure 6, can be computed manually, as shown in figure 7 and figure 8.

Let “Paraci” be inputted characters, and word in the database is Paraco.

m = Inputted by user = Paraci
n = Word in the database = Paraco

d[0,0] = 0

Initialize first row and first column with 0,1,2,. . . m 0,1,2,...n

		P	A	R	A	C	O
	0	1	2	3	4	5	6
P	1	0					
A	2	1					
R	3	2					
A	4	3					
C	5	4					
I	6	5					

Figure 7. The First row and column initialization

- For each character, compare each character from inputted word with an actual word in the database. If it is a match, then the cost is 0. Otherwise the cost will be 1
- Check the minimum,

  d[i,j] Top = d[i,j]+1
  Side = d[i,j]+1
  Diagonal = d[i,j]+ cost
- Compare character P with P, put cost = 1 if differ, otherwise cost =0

  Check all values in d [i,j] Top = 1, Diagonal = 0, Side = Minimum diagonal
- d [i,j] = d [i,j] + cost = 0 + 0 = 0 so on, so forth

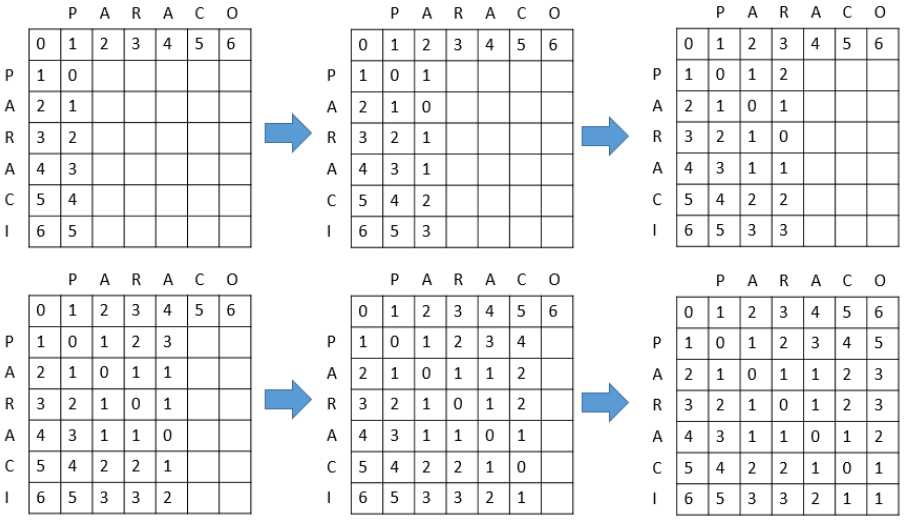

Figure 8. Manual computation process of The *Levenshtein Distance* algorithm

In Figure 8, the distance generated is a value that is in the lower-right corner of the matrix, which is 1. The value of one means there is 1 operation performed. The value of one is generated from the operation of the sum of the cost values with a minimum diagonal value. The distance value obtained from the diagonal side means that the operation that works is a

substitution. So for the PARACI String to be converted into a PARACO string, one operation is needed, namely the substitution of the first character ("I") to the character "O" so that the value of the Levenshtein Distance is equal to 1. The result of the Levenshtein Distance algorithm implementation on drugs e-dictionary, as depicted in figure 9.

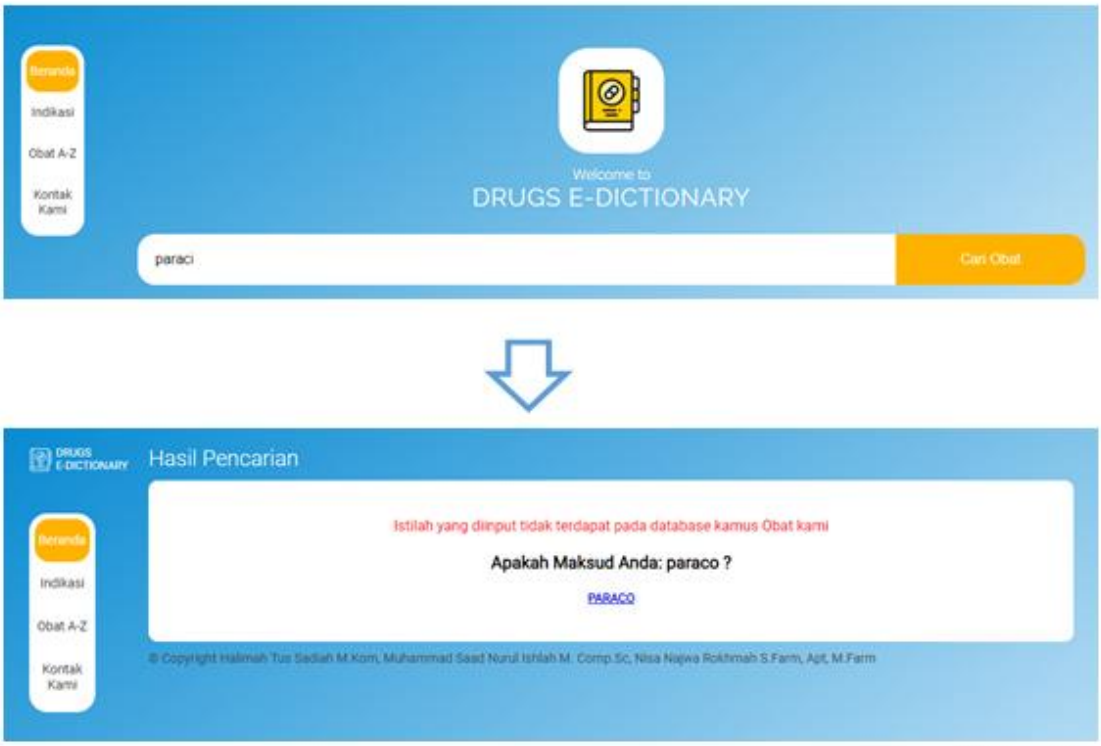

Figure 9. Result of Levenshtein Distance algorithm implementation on drugs e-dictionary

3.2.1. Testing the Drugs e-Dictionary with Query Suggestion Added Facility

Tests carried out in the form of validation testing by inputting 100 test queries into the search form. Table 1 summarizes the results of the validation test on the drug e-dictionary. Figure 9 shows an example of query suggestion testing.

Table 1. Example of Words in query suggestion validation testing

No	Inputted Query (drug name)	*Query Suggestion*	*Output*	Category of Levenshtein Distance Algorithm Operation	Notes	Validation
1	Paracetamol	-	Paracetamol	-	The inputted query is correct	Valid
2	Kamols	*"Apakah maksud anda kamolas, ?"*	Kamolas	Add letter a	Incorrect query inputted, lacking letters	Valid
3	Zephanall	*"Apakah maksud anda Zephanal?"*	Zephanal	delete letter l	Incorrect query entered, Excess	Valid

No	Inputted Query (drug name)	*Query Suggestion*	*Output*	Category of Levenshtein Distance Algorithm Operation	Notes	Validation
4	Paraci	*"Apakah maksud anda paraco?"*	Paraco	Substitute i with o	letters Incorrect query entered	Valid
5	Diparin	*"Apakah maksud anda Dapyrin ?"*	Dapyrin	*Query suggestion* by closest word	The inputted query does not exist in the database	Valid

In Table 1, validation tests are categorized into an insert, delete, and substitution operations. Whenever an inputted query is not in the database, the system will show notification of "The inputted query does not exist in the database", which then will show *Query suggestion* by generating some terms that are closer in the database. Let's take "diparin" as an inputted query (unknown term in the database). The system will show "dapyrin" as the suggestion (Table 1).

The developed system uses a non-case sensitive query checking. Hence it will not affect the output, whether the inputted query is in an uppercase or lowercase. In addition, if the inputted query is a meaningless word, such as "ZZZZ", then the system will show a word with initial letter Z that has the fewest number of words in the database, in this case "Zalona". The system will search for any terms with minimal Levenshtein Distance algorithm operation.

The evaluation of accuracy represented in a confusion matrix (Table 2), which has four classification process results, namely: *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN) [20].

Table 2. Confusion Matrix

Total Population	Predicted: yes	Predicted: no
Actual: yes	TP	FN
Actual: no	FP	TN

Based on TN, FP, FN and TP accuracy are obtained (equation 1), precision (equation 2) and *recall* (equation 3). Based on equation 1, equation 2 and equation 3, we have a result of query accuracy for drug query of 90%, precsion=90%, recall = 90%. The confusion matrix for the evaluation result of drug terms is in Table 3.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} * 100\% \quad (1)$$

$$presisi = \frac{TP}{TP + FP} * 100\% \quad (2)$$

$$recall = \frac{TP}{TP + FN} * 100\% \quad (3)$$

Table 3. Confusion Matrix for Drug Terms Evaluation

100	Predicted: yes	Predicted: no
Actual: yes	45	5
Actual: no	5	45

4. Conclusion

The drug e-dictionary search function needs to be optimized with the addition of the Query Suggestion facility. The Query Suggestion facility was developed using the Levenshtein Distance algorithm. Based on the results of the implementation, the Levenshtein Distance algorithm runs from the top left corner of a two-dimensional array that has been filled with several initial string characters and target strings and is given a cost value. The cost value at the lower right-hand end is the Distance edit value that represents the number of operations that the algorithm has to process. Based on the test results, the system can evaluate words that are not in the database with the query suggestion function closest to the database. It reaches 90% accuracy of the inputted query, with 90% precision and 90% recall in the confusion matrix. The future work is the implementation of n-gram on drugs e-dictionary and performing a comparative analysis of Levenshtein distance algorithm with n-gram.

References

[1] Departemen Kesehatan RI. Tanggung Jawab Apoteker Terhadap Keselamatan Pasien (Patient Safety). Jakarta: Direktorat Bina Farmasi Komunitas Dan Klinik Ditjen Bina Kefarmasian Dan Alat Kesehatan Departemen Kesehatan RI. 2008.

[2] Y. Song, & Li-wei He. 2010. Optimal Rare Query Suggestion With Implicit User. ACM Journals.pp: 901-910.

[3] S. Jiang, S. Zilles, & R. Holte. 2008. Query suggestion by query search: a new approach to user support in web search [Online]. [Cited 2018 August 1]. Available from www.cs.uregina.ca/~zilles/jiangZH09.pdf

[4] Y. Song, D. Zhou., & L.W. He. 2011. Post-ranking-query-suggestion-by-diversifying-searchresul [Online]. [Cited 2018 August 1]. Available from https://www.microsoft.com/id-id/: https://www.microsoft.com/en-us/research/publication/post-ranking-query-suggestionbydiversifying-search-results/

[5] J.-M.Yangy, R. Cai, F. Jingz, S.Wangy, L. Zhangy, & W.Y.Ma. 2008. Search-based Query Suggestion.[Online] [Cited 2018 August 1]. Available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.159.3499&rep=rep1&type=pdf

[6] Q. Mei, D. Zhou & K Church 2008. Query Suggestion Using Hitting Time [Online]. [Cited 2018 August 1]. Available from https://www.microsoft.com/enus/ research/wpcontent/uploads/2017/01/sugg.pdf

[7] H. Cao, D. Jiang,J. Pei, Q. He, Z. Liao, E. Chen, & H. Li. 2008. Context-Aware Query Suggestion by Mining Click-Through [Online]. [Cited 2018 August 1]. Available from https://www.cs.sfu.ca/~jpei/publications/QuerySuggestion-KDD08.pdf

[8] Z.-J. Zha, L. Yang, T. Me., M. Wang, & Zengfu. Visual Query Suggestion. *ACM Journals*. pp. 15-24. 2009.

[9] S. Bathia, D. Majumdar, & P. Mitra. Query Suggestions in the Absence of Query Logs. *ACM Journals*, pp. 1-10.2011.

[10] K.N. Ngafidin & H. Wibawanto. Implementasi Fitur Autocomplete dan Algoritma Levenshtein Distance untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI). *Jurnal Teknik Elektro*. Vol. 7, No. 1, pp.1-6. 2015

[11] R Pressman dan B.R. Maxim. Software Engineering a Practitioners approach. McGraw-Hill Education : New York. 2014.

[12] J Satzinger, R. Jackson, & S. Burd. System Analysis and Design in a changing World. USA: Course Technology Cengage Learning. 2010.

[13] Ikatan Apoteker Indonesia. ISO Informasi Spesialite Obat Indonesia. Vol 52. 2019. Jakarta : Isfi Penerbitan.2019

[14] Z.Afriansyah, D.Puspitaningrum, & Ernawati. Rancang Bangun Aplikasi Pencocokan DNA Manusia Menggunakan Algoritma Levenshtein Distance (Studi Kasus: Dna Kanker Hati Manusia). *Jurnal Rekursif* . Vol. 3, No. 2,pp. 61-67.2015.

[15] B. Pratama & S. Pamungkas, Analisis Kinerja Algoritma Levenshtein Distance Dalam Mendeteksi Kemiripan Dokumen Teks. *Jurnal Log!k@* . Vol. 6, No. 2, pp. 131-143.2016

[16] T. Aprilianto, & A. Badawi. Sistem Koreksi Kata Dan Pengenalan Struktur Kalimat Berbahasa Indonesia Dengan Pendekatan Kamus Berbasis Levenshtein Distance. *Jurnal SPIRIT*. Vol. 9, No. 1, pp 48-61. 2017.

[17] R. Haldar, & D. Mukhopadhyay. 2011. Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach [Online]. [Cited 2018 August 1]. Available from

[18] R. Mishra, & N. Kaur. A Survey of Spelling Error Detection and Correction Techniques. *International Journal of Computer Trends and Technology*. Vol. 3, No. 4, pp. 372-374. 2013

[19] N. Ariyani, N., R. Sutardi, & Ramadhan. Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance. *semanTIK*.Vol. 2, No. 1,pp. 279-286. 2016.

[20] M. Navin, Pankaja R. Performance Analysis of Text Classification Algorithms using Confusion Matrix. *International Journal of Engineering and Technical Research (IJETR)*. Vol. 6, No. 2,pp. 75-78. 2016