Scientific Journal of Informatics
Vol. 5, No. 1, May 2018

p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Autocomplete and Spell Checking Levenshtein Distance Algorithm to Getting Text Suggest Error Data Searching in Library

Muhammad Maulana Yulianto[1], **Riza Arifudin**[2], **Alamsyah**[3]

[1,2,3] Computer Science Departmement, Natural Sciences, State University of Semarang
Email: [1]yuliantoolan@gmail.com, [2]rizaarifudin@mail.unnes.ac.id, [3]alamsyah@mail.unnes.ac.id

Abstract

Nowadays internet technology provide more convenience for searching information on a daily. Users are allowed to find and publish their resources on the internet using search engine. Search engine is a computer program designed to facilitate a user to find the information or data that they need. Search engines generally find the data based on keywords they entered, therefore a lot of case when the user can't find the data that they need because there are an error while entering a keyword. Thats why a search engine with the ability to detect the entered words is required so the error can be avoided while we search the data. The feature that used to provide the text suggestion is autocomplete and spell checking using Levenshtein distance algorithm. The purpose of this research is to apply the autocomplete feature and spell checking with Levenshtein distance algorithm to get text suggestion in an error data searching in library and determine the level of accuracy on data search trials. This research using 1155 data obtained from UNNES Library. The variables are the input process and the classification of books. The accuracy of Levenshtein algorithm is 86% based on 1055 source case and 100 target case.

Keywords: Autocomplete, Spell Checking, Levenshtein Distance, Library Automation, Data Searching.

1. INTRODUCTION

Many researchers use the Web search engines' hit count as an estimator of the Web information distribution in a variety of knowledge-based (linguistic) tasks [1]. The Web has tremendous collection of useful information however, extracting the accurate information from the web is extremely difficult, because the current search engines are restricted to the keyword-based search techniques. Search engines generally find the data based on keywords they entered, therefore a lot of case when the user cannot find the data that they need because there are an error while entering a keyword. Actually, the web has tremendous collection of useful information however, extracting the accurate information from the web is extremely difficult, because the current search engines are restricted to the keyword-based search techniques. Thus, the interpretation of information contained in web documents is left to the human user to done manually [2]. Autocomplete is a feature that provided by many web browsers such a search engines interface, word processor, and command line interpreter. Morover, it's able to give advice for users without writting the whole word completely [3]. Approximate string matching (ASM) is a well-known computational problem with important applications in database searching, plagiarism detection, spelling correction, and bioinformatics. Levenshtein distance algorithm is one of Approximate string matching algorithm used in search string based on the estimation approach [4]. This algorithm is a weighting approach to appoint a cost of 1 to every edit operations (Insertion, deletion and substitution). This distance is known as Levenshtein distance, a special case of edit distance where unit costs apply [5]. By using this algorithm, the searching perform is more accurate. Even the entered word is incorrect, this algorithm still can find the data that user needs and provide search suggestions approaching from the word entered [6]. On previous research, the Levenshtein distance algorithm gives good results in resolving problems in matching the string data to provide text suggest, for example, in handwriting recognition, search words and misspelled words so that the effectiveness of inputting increases, spelling mistakes can be avoided and autocomplete speeds up human computer interactions [7].

Autocomplete to provide choice word suggestions while typing is a feature provided in the browser, search engines, word processing, or on the command line. Besides these features are able to give advice that users need without having to be written as a whole [3]. Automatic spell checker systems aim to verify and correct erroneous words through a suggested set of words that are the nearest lexically to the erroneous ones [8]. The spell check can be divided into two main subproblems: the error verification and the correction of errors found. The first one is simply to determine whether the word belongs to the target language. The process may involve a morphological analysis of the word and statistical approaches, or just check the existence of this word in a dictionary that represents almost completely the target language [9]. With the use of both of these features, the system provides text suggest the user during the input process and after the input process.

Library automation has contributed in terms of membership data processing, circulation and cataloging [10]. To realize an increase in such services, UNNES Library since 2009 has utilized information technology services such as library automation [11]. Library automation has the following objectives, namely [12] Simplify, accelerate, and the correct service, speed up information searches with good results, speed up the discovery of information by the users themselves through the facility search was mounted (online) and expanding outreach to diverse sources of information. Based on the above, the purpose of this research is to apply the autocomplete feature and spell checking using Levenshtein distance algorithm to determine the accuracy of the search obtained from the text suggest the data search error in the library.

2. METHODS

2.1. Levenshtein Distance

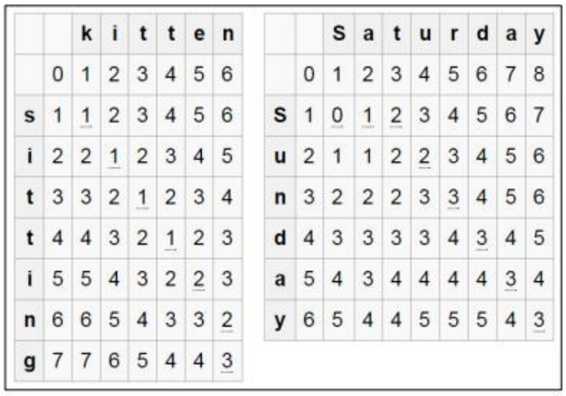
Levenshtein distance algorithm invented by Vladimir Levenshtein, a scientist from Russia in 1965 [13]. Levenshtein Distance between any two strings is defined as the minimum number of edits that we can do to transform one string into another [14]. Levenshtein distance algorithm is particularly useful in applications that are used to determine the similarity of two strings [15]. Levenshtein distance algorithm is an algorithm editor that utilizes dynamic programming string for operation. Levenshtein distance is the minimum distance required to transform one string to another string [7]. This algorithm is a weighting approach to appoint a cost of 1 to every edit operations (Insertion, deletion and substitution). For instance, the Levenshtein edit distance between “dog” and “cat” is 3 (substituting d by c, o by a, g by t) [16]. The Levenshtein algorithm helps define the number of modifications; insertions and deletions in a string c1 to be the same as a string c2. It calculates the minimum number of elementary operations executed. To this end, the algorithm uses a matrix of (n+1)*(m+1) dimension, where n and m are the two previous strings’ lengths. The calculation of the cell M[N,P] is equal to the minimum value between the executed elementary operations [8].

$$M(i,j) = Min \begin{cases} M(i, j-1)+1) \\ M\big((i-1, j)+1\big) \\ M\big((i-1, j-1)+(i-1, j-1\big) \end{cases} \tag{1}$$

where

$$\begin{aligned} (i,j) &= 0 \text{ if } \mathrm{T}(i) = \mathrm{S}(j) \\ &= 1 \text{ if } \mathrm{T}(i) \neq \mathrm{S}(j) \end{aligned} \tag{2}$$

There are three kinds of major operations that can be performed by this algorithm, namely character insertion operation to add a certain character into a string, then character substitution operation to replace a specific character in a string, and character deletion operation to remove the particular character in a string.
To calculate the distance with Levenshtein distance can be done using a table as shown in Figure 1. In Figure 1 there is a cell with a number of letters that will be checked. Charging the table starting from cell (1,1) with the position of the rows as the index i and column position as the index j. Charging table based Levenshtein distance algorithm [17] is shown in Figure 1.



		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Figure 1. Levenshtein distance calculation table

2.2. Design process

The steps to create a library automation by applying autocomplete and spell checking using Levenshtein distance algorithm is shown in Figure 2.

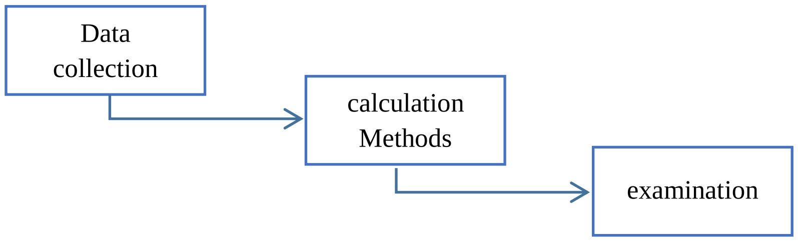


Figure 2. The steps to create library automation

2.3. System planning

This expert system design using the waterfall model. Waterfall is an approach and based on the assumption that big decisions have to be made before coding begins [18]. There are four stages in the waterfall which is a requirement analysis, design, implementation and testing [19]. First step is requirement gathering and analysis to defining the entire software format, identifying all the needs, and outlines of the created system [20]. Then second step is execute design applications including interface design, and database structure design [21]. To create an interesting web-based application program (website) it must be designed beforehand, so the achieved results are suitable with the predetermined objectives [22]. Third step is implementation to relized designing software as a series of program or program unit [23]. The last step is testing. To test whether the system is ready and feasible to use. The tester can define the set of input conditions and perform testing on functional specifications of the program [24].

3. RESULTS AND DISCUSSION

3.1. Data Collecting

Data used in the form of Unnes Library Unit data book of the year 2000-2017 covering the Central Library, Faculty of Language and Art, Faculty of Mathematics and Natural Sciences, Faculty of Law, Faculty of Engineering, Faculty of Economics, Faculty of Social Sciences, Faculty of Education, and the Graduate UNNES, with the amount of data as much as in 1155 the book's title and a description of the book.

3.2 Autocomplete

Autocomplete [25] is a feature provided by many web browsers, e-mail programs, search engine interfaces, source code editors, database query tools, word processors, and command line interpreters. Autocomplete aims to obtain an appropriate book recommendations by input to a minimum. To get started with a book recommendation takes input from a field provided (string source) and retrieve data from the database (string target). then matched using Levenshtein algorithm to create a matrix along the 0 ... *m* rows (string source) and 0 ... *n* columns (string target). Then fill in the entry [*m*, *n*] of [*i*, *j*] with the initialization of the matrix. If the string value equal to the string *i*, *j* then obtained his distance value of 0 for the value of the source string is equal to the target string. Then the data book that has snippets of words to appear as book recommendations. Flowchart autocomplete can be seen in Figure 3.

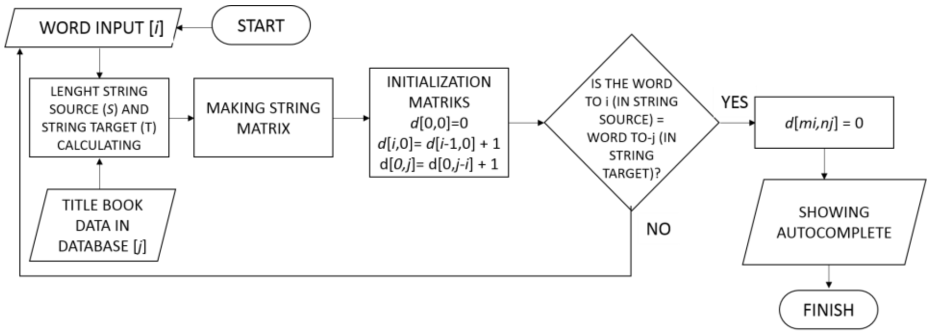

START
WORD INPUT [i]
LENGHT STRING SOURCE (S) AND STRING TARGET (T) CALCULATING
TITLE BOOK DATA IN DATABASE [j]
MAKING STRING MATRIX
INITIALIZATION MATRIKS d[0,0]=0 d[i,0]= d[i-1,0] + 1 d[0,j]= d[0,j-i] + 1
IS THE WORD TO i (IN STRING SOURCE) = WORD TO-j (IN STRING TARGET)?
YES
NO
d[mi,nj] = 0
SHOWING AUTOCOMPLETE
FINISH

Figure 3. Flowchart autocomplete

3.3 Spell Checking

Detecting the word error can be done by using a computer aided application. The application used to detect and handling word error is called spelling checker [26]. A spell checker is a technique which identifies the incorrect or misspelled words and replaces them with the best possible combination of correct words [27]. Spell checking aims to obtain an appropriate book recommendations with advice (autocorrect) when the user performs inputting error. To obtain such advice starts with taking input from a field provided (string source) and retrieve data from the database (string target). then matched using Levenshtein algorithm to create a matrix along the 0 ... *m* row (string source) and 0 ... *n* columns (string target). Then fill in the entry [*m*, *n*] of [*i*, *j*] with the initialization of the matrix. If the string value equal to the string *i*, *j* then obtained his distance value of 0 for the value of the source string is equal to the target string and bring autocomplete. Then, if the value of the string i is not equal to the string j then to fill in the entry [*i*, *j*] is to find the minimum value of *d* [*i*, *j*-1], *d* [*i*-1, *j*] and *d* [*i*-1, *j*-1]. After getting the distance value of 0 <[*mi*, *nj*] <= 2 then the system suggest improvements word with the smallest distance in accordance with the existing data in the database. Flowchart spell checking can be seen in Figure 4.

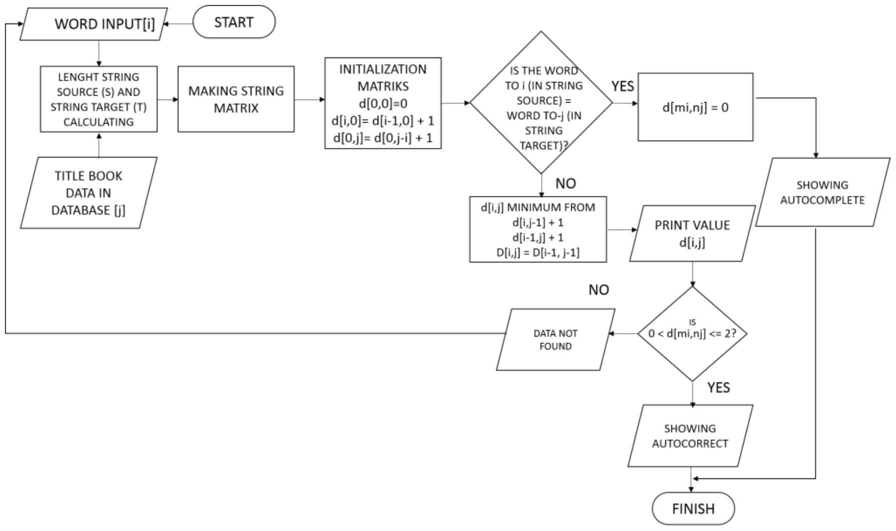

START
WORD INPUT[i]
LENGHT STRING SOURCE (S) AND STRING TARGET (T) CALCULATING
TITLE BOOK DATA IN DATABASE [j]
MAKING STRING MATRIX
INITIALIZATION MATRIKS d[0,0]=0 d[i,0]= d[i-1,0] + 1 d[0,j]= d[0,j-i] + 1
IS THE WORD TO i (IN STRING SOURCE) = WORD TO-j (IN STRING TARGET)?
YES
d[mi,nj] = 0
SHOWING AUTOCOMPLETE
NO
d[i,j] MINIMUM FROM d[i,j-1] + 1 d[i-1,j] + 1 D[i,j] = D[i-1, j-1]
PRINT VALUE d[i,j]
IS 0 < d[mi,nj] <= 2?
NO
DATA NOT FOUND
YES
SHOWING AUTOCORRECT
FINISH

Figure 4. Flowchart autocorrect

3.4. Calculation Methods

1). Calculation of the distance to the target string "saturday" and the source string "sunday", can be seen in Table 1.

Table 1. Levenshtein operation with the source string sunday

Index	1	2	3	4	5	6	7	8
Target	s	a	t	u	r	d	a	y
Source	s	u	n	d	a	y		

2). Showing autocorrect. With the data obtained in Table 1, then the formula is obtained,

$$\begin{aligned} D(s,t) &= \sum_{i=1}^{8} d(s_i, t_i) \\ &= d(s_1, t_1) + d(s_2, t_2) + d(s_3, t_3) + d(s_4, t_4) + d(s_5, t_5) + d(s_6, t_6) + d(s_7, t_7) + d(s_8, t_8) \\ &= d(s, s) + d(u, a) + d(n, t) + d(d, u) + d(a, r) + d(y, d) + d(-, a) + d(-, y) \\ &= 0 + 1 + 1 + 1 + 1 + 1 + 1 + 1 \\ &= 7 \end{aligned}$$

3). The result of levenshtein distance with string target "saturday" and the source string "sunday" is 7.

3.5. Testing

From the application of 100 data as a target of a case (test data) and in 1055 as a source case (training data), after experimenting with 3-way operation algorithm Levenshtein includes Deletion, Insertion and substitution, gained 86 data that can display on the appropriate word as it looks in Table 2.

Table 2. Test Result Data

Number of test data	Data corresponding	The data do not match	accuracy
100	86	14	86%

From the test results data using a system that has been created, resulting in as many as 86 test data in accordance with the desired data and 14 data that is not appropriate because the system is searching for data with a minimum value, where the trials are the input data is wrong but it is worth the same as the existing data in the database, so the system reads the data as correct data.

4. CONCLUSION

The process of implementing the autocomplete and spell checking of data contained in the search process. In the search process data using Levenshtein distance algorithm that has three string matching operation includes the deletion, insertion and substitution. Research results obtained suggest is the appearance of text on the search system includes autocomplete and autocorrect. Spell checking accuracy rate obtained from the system by 86%, calculated using 1055 data as a source case (training data) and 100 data as a target of a case (test data).

5. REFERENCES

[1] Martínez, L.S., & Sánchez, D. (2016). Evaluating the suitability of Web search engines as proxies for knowledge discovery from the Web. *Procedia Computer Science*, 96, 169-178.

[2] Awny, S., & Amal A, M. (2017). IBRI-CASONTO: Ontology-based semantic search engine. *Egyptian Informatics Journal*, 18, 181-192.

[3] Yao, Z. (2013). Implementation Of The Autocomplete Feature Of The Textbox Based On Ajax And Web Service. *Journal of Computers*, (8) 9 : 2197-2203.

[4] Rubio, M. 2015. A Consensus Algorithm for Approximate String Matching. *Iberoamerican Conference on Electronics Engoneering and Computer Science*, 7, 322-327.

[5] Ayad, L. A., Barton, C., & Pissis, S. P. (2017). A faster and more accurate heuristic for cyclic edit distance computation. *Pattern Recognition Letters*, *88*, 81-87.

[6] Umar, R., Hendriana, Y., & Budiyono, E. (2015). Implementation of Levenshtein Distance Algorithm for E-Commerce of Bravoisitees Distro. *International Journal of Computer Trends and Technology (IJCTT),* 27 (3), 131-136.

[7] Putra, M. E. W., & Suwardi, I. S. (2015). Structural off-line handwriting character recognition using approximate subgraph matching and levenshtein distance. *Procedia Computer Science*, *59*, 340-349.

[8] Nejja, M. & Yousfi, A. (2015). The Context In Automatic Spell Correction. *The International Conference on Advanced Wireless, Information, and Communication Technologies (AWICT),* 73, 109-114.

[9] Andrade, G., Teixeira, F., Xavier, C. R., Oliveira, R. S., Rocha, L. C., & Evsukoff, A. G. (2012). HASCH: high performance automatic spell checker for Portuguese texts from the web. *Procedia Computer Science, 9*, 403-411.

[10] Yudie, I., Mustafid, P. & Sugiharto, A. (2011). Perancangan Sistem Informasi Perpustakaan Berbasis Web Application. *Jurnal Sistem Informasi Bisnis,* 01, 70-73.

[11] Yudhistira, E., Purwinarko, A., & Wusqo, I. U. (2016). Implementasi Restful Web Service Menggunakan AsyncTask pada Aplikasi Library Automation Berbasis Android. *Seminar Nasional Ilmu Komputer (SNIK 2016),* 286-292.

[12] Nurendah, Y., & Mulyana, M. (2013). Analisis Pengaruh Kualitas Pelayanan Perpustakaan Terhadap Kepuasan dan Hubungannya dengan Loyalitas Mahasiswa. *Jurnal Ilmiah Manajemen Kesatuan,* 1(1): 93-112.

[13] Janowski, T., & Mohanti, H. (2010). *Distributed Computing and Internet Technology*. India: Springer.

[14] Desai, N. & Narvekar M. (2015). Normalization of Noisy Text Data. *International Conference on Advanced Computing Technologies and Applications (ICACTA),* 45, 127-132.

[15] Mary, R., Nishikant, A. S., & Iyengar, N. C. S. (2014). Use of Edit Distance Algorithm to Search a Keyword in Cloud Environment. *International Journal of Database Theory and Application*, *7*(6), 223-232.

[16] Mishra, R., & Kaur, N. (2013). A Survey of Spelling Error Detection and Correction Techniques. *International Journal of Computer Trends and Technology*, *4*(3), 372-374.

[17] Chowdhury, S.D., Bhattacharya U., & Parui S.K. (2013). Online Handwriting Recognition Using Levenshtein Distance Metric. *Document Analysis and Recognition (ICDAR),*79-83.

[18] Patel, U.A., & Jain N.K. (2013). New Idea in Waterfall Model for Real Time Software Development. *International Journal of Engineering Research & Technology (IJERT),* 2(4), 115.

[19] Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

[20] Nugroho, Z. A., & Arifudin, R. (2015). Sistem Informasi Tracer Study Alumni Universitas Negeri Semarang Dengan Aplikasi Digital Maps. *Scientific Journal of Informatics, 1*(2), 153-160.

[21] Putra, A. T. (2015). Pengembangan E-Lecture menggunakan Web Service Sikadu untuk Mendukung Perkuliahan di Universitas Negeri Semarang. *Scientific Journal of Informatics, 1*(2), 168-176.

[22] Vedayoko, L. G., Sugiharti, E., & Muslim, M. A. (2017). Expert System Diagnosis of Bowel Disease Using Case Based Reasoning with Nearest Neighbor Algorithm. *Scientific Journal of Informatics, 4*(2), 134-142.

[23] Purwinarko, A., & Sukestiyarno, Y. L. (2015). Model Expertise Management System di Universitas Negeri Semarang. *Scientific Journal of Informatics, 1*(2), 177-184.

[24] Mustaqbal, M. S., Firdaus, R. F., & Rahmadi, H. (2016). Pengujian Aplikasi Menggunakan Black Box Testing Boundary Value Analysis (Studi Kasus: Aplikasi Prediksi Kelulusan SMNPTN). *Jurnal Ilmiah Teknologi Informasi Terapan*, *1*(3), 31-36.

[25] Bo, W., & Han-bo, W. (2010). Implementation of Auto Complete Based on

Jquery. *Journal of SanMenXia Polytechnic*, *9*(3), 102-126.

[26] Kamayani, M., Reinanda, R., Simbolon, S., Soleh, M. Y., & Purwarianti, A. (2011). Application of document spelling checker for Bahasa Indonesia. *Advanced Computer Science and Information System (ICACSIS)*, 249-252.

[27] Kaur, A., Singh, P., & Rani, S. (2014). Spell Checking and Error Correcting System for text paragraphs written in Punjabi Language using Hybrid approach. *International Journal of Engineering and Computer Science*, *3*(09), 8030-8032.