Fuzzy string implementation matching on android-based encyclopedia and anatomy quiz

M Firdaus’, M H Mud’is, I Nurjaman, I Fauzia and B Busro
UIN Sunan Gunung Djati Bandung, Jl. AH Nasution No. 105 Bandung, West Java, Indonesia

*maulanafirdaus2014@gmail.com

Abstract. Encyclopedia anatomy is an android application to introduce anatomy, equipped with a search feature, multiple choice quizzes and guess image quizzes. Due to some anatomical term is uptake from foreign languages and frequently typing errors by a smartphone user, then search feature in this application implement fuzzy string matching method which uses Levenshtein distance algorithm. However, the results obtained are still not accurate, then added method of splitting and indexing, so the results are more accurate and search process fast enough. Due to more simple than a book and supported by a good appearance, this application can get a positive response from the users.

1. Introduction
Anatomy (derived from Greek anatomy or anatemnein, which means cutting) is a branch of biology that deals with the structure and organization of living things. Anatomy is the knowledge needed for students and practitioners in the field of biology and medicine to know the basics of the structure of living things both human, animal and plant [1].

Some terms in anatomy are absorption from foreign languages so that it is quite difficult to learn [2-6]. The development of technology, especially smartphones, enables the creation of a more practical and interesting media for introducing anatomy.

The highly sophisticated smartphone technology apparently cannot solve all problems; it is still found that there is a possibility of user errors / human errors such as typographical errors (typo) [7]. The current system uses exact string matching so that it cannot overcome the problem of typing errors by the user which results in the information obtained is not as expected, or even information cannot be found.

A string matching method is needed that can resolve typing errors so that the desired information can appear even though there are a few typing errors. Based on the background of the problems described, the authors are interested in making a study of the implementation of fuzzy string matching in an encyclopedia and android based anatomy quiz.

2. Methodology
The data used is taken from the class VIII Integrated IPA Electronic School Book [8] and the Biology Electronic School Book class XI [9], and the data taken is only about the anatomy of the human body. Then the data is entered into the SQLite database so that the application can run offline. The algorithm applied in this research is the Levenshtein distance algorithm which is one of the fuzzy string matching algorithms [10].
This algorithm runs from the top left corner of a two-dimensional array that has been filled with several source characters and target strings and is given a cost value. The cost value at the lower right end becomes the edit distance value which describes the number of differences in two strings.

Calculations in this algorithm include insert operations, delete and substitution characters in the source string to calculate a distance. The description of the matrix calculation can be seen in the figure1.

| String Source = Bronkus |
|-------------------------|
| String Target = Bronkus |

|   | b | r | o | n | k | u | s |
|---|---|---|---|---|---|---|---|
| b | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| r | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| o | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| n | 3 | 2 | 1 | 0 | 1 | 2 | 3 |
| k | 4 | 3 | 2 | 1 | 0 | 1 | 2 |
| u | 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| s | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

MaxLength = 7
Dis = 0

*Similarity Value (SV)*

\[ SV = \{1-(\text{Dis}/(\text{MaxLength}(CS,ST))}\]  
\[ SV = \{1-(0/7)\} \]

\[ SV = 1 \]

3. Results and discussion

In this study fuzzy string matching is implemented in the anatomical name search feature using the Levenshtein Distance algorithm. To be able to provide the best solution to the background of this research problem in the form of frequent typing errors, preprocessing is implemented in the form of splitting and indexing methods, and after processing in the form of ranking / shorting.

So that in this study three search methods will be applied, namely Levenshtein Distance Standard, Levenshtein Distance with splitting, and Levenshtein Distance with splitting and indexing. The three methods will be tested to find out which method can display the best search results and applied as the main search method on the application.

3.1. Implementation of Levenshtein Distance standard

Figure 1 is the implementation of the Levenshtein Distance algorithm in the Java programming language which functions to calculate the similarity between two strings, if the result is close to 0 (zero) it means the string has no resemblance and if the result is close to 1 (one) means the string has a high resemblance. The standard Levenshtein distance algorithm performs string matching without doing any process before comparing the keyword string entered by the user with each anatomic name string.
3.2. Implementation of Levenshtein Distance with splitting

```java
public static double similarity(String s1, String s2) {
    String longer = s1, shorter = s2;
    if (s1.length() < s2.length()) {
        longer = s2;
        shorter = s1;
    }
    int longerLength = longer.length();
    if (longerLength == 0) {
        return 1.0;
    }
    return (longerLength - editDistance(longer, shorter)) / (double) longerLength;
}

public static int editDistance(String s1, String s2) {
    s1 = s1.toLowerCase();
    s2 = s2.toLowerCase();
    int[] costs = new int[s2.length() + 1];
    for (int i = 0; i <= s1.length(); i++) {
        int lastValue = 1;
        for (int j = 0; j <= s2.length(); j++) {
            if (i == 0) {
                costs[j] = j;
                break;
            } else {
                int newValue = costs[j - 1];
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    newValue = Math.min(Math.min(newValue, lastValue), costs[j]);
                    costs[j] = 1;
                } else {
                    newValue += 1;
                    costs[j] = newValue;
                }
            }
        }
        lastValue = costs[s2.length()];
    }
    return costs[s2.length()];
}
```

**Figure 2.** Levenshtein Distance standard.

**Figure 3.** Source code splitting.
Figure 3 is a search implementation using the pre-processing splitting method. Before the keyword string is compared to the anatomical name strings, the spacing will be separated if the string has more than one word. For example, the string "Spine" consists of two words, then the separation is based on space into the string "Bone" and the string "Back" after that the string has been separated compared to the keyword.

3.3. Implementation of Levenshtein Distance with splitting and indexing

```java
        cari = editText.getText().toString();
        cari = cari.toLowerCase();
        String st = cari;
        String[] splitSt = st.split("\\s");
        for (j = 0; j < listKonten.length; j++) {
            double all = 0, part1 = 0, part2 = 0, temp = 0;
            pembanding = listKonten[j].toLowerCase();
            if (cari.charAt(0) == pembanding.charAt(0))
                all = similarity(cari, pembanding);
            String str = pembanding;
            String[] splitStr = str.split("\\s");
            for (int i = 0; i < splitSt.length; i++) {
                for (int j = 0; j < splitStr.length; j++) {
                    if (splitStr[i].charAt(0) == splitStr[j].charAt(0))
                        part2 = similarity(splitSt[i], splitStr[j]);
                }
            }
            if (all <= part1) {
                temp = part1;
            } else {
                temp = all;
            }
            if (temp >= 0.5f) {
                Tampil[jumlah][0] = listKonten[j];
                Tampil[jumlah][1] = String.format("%.2f", part1);
                Tampil[jumlah][2] = String.format("%.2f", all);
                jumlah++;
            }
        }
```

Figure 4. Implementation of Levenshtein Distance with splitting and indexing.

Figure 4 is a search implementation using the preprocessing splitting method like the third method, only the indexing method is added, so the system only searches if the first character in the keyword is the same as the first character in the names of anatomy.

3.4. Accuracy testing of the Levenshtein Distance algorithm

Accuracy test results on the three Levenshtein distance methods, where testing is done 36 times that have been adjusted, if the target string is found then given a value of 1 and if the target string is not found then given a value of 0. Then the test results for each method are summed and divided by the number of tests performed, so that the average accuracy value of each method is obtained.
The test results show that Levenshtein Distance Standard has an accuracy of 53%, Levenshtein Distance with splitting has an accuracy of 100%, and Levenshtein Distance with splitting and indexing has an accuracy of 94%. Furthermore, to further clarify the test results, the test data is displayed in figure 5.

![Accuracy Graph](image)

**Figure 5.** Accuracy graph.

### 3.5. Execution Speed Test Levenshtein Distance algorithm

The results of the search process speed test on the three methods of the Levenshtein distance, where testing is done 36 times adjusted, the time unit used in this study are milliseconds / milliseconds. Furthermore, the length of the search time for each method is summed and divided by the number of tests performed, so that the average speed of each method is obtained. The test results show that Levenshtein Distance Standard has average speed of 10.97 milliseconds, Levenshtein Distance with splitting has average speed of 80.5 milliseconds, and Levenshtein Distance with splitting and indexing has average speed of 44.61 milliseconds. Furthermore, to further clarify the test results, the test data is displayed in the graph in the figure 6 and 7.

![Process Speed Graph](image)

**Figure 6.** Process speed graph.

![Average Process Speed Graph](image)

**Figure 7.** Average process speed graph (ms).

### 3.6. System feasibility test

The system feasibility test aims to get a direct assessment from the respondents on the system produced. The sample in this study amounted to 50 people. The samples in the feasibility test of this system are students majoring in biology and science biology.

The questionnaire in this study contains an assessment of the appearance, ease of use, system performance, content or application content. The results of the feasibility test show that the average value of the display category is 86.625%, ease of use 93.75%, system performance 85.5% and
content/content 81%. Based on the assessment category, it can be concluded that the display categories include "Very Good," ease of use including "Very Good," System Performance including "Very Good" and content/content also including "Very Good."

4. Conclusion
This encyclopedia application and anatomical quiz are classified into very good categories; the average display is 86.625%, ease of use 93.75%, system performance 85.5% and content/content 81%. Fuzzy string matching method using the Levenshtein distance algorithm, in general, can overcome typing errors but has a weakness in terms of accuracy of only 53%. So to be able to display better search results, in this study the Levenshtein distance algorithm is combined with a splitting method that produces 100% accuracy except that the search time becomes longer. So to slightly speed up the search process, an indexing method is added, resulting in the performance of the Levenshtein distance algorithm with high accuracy (94%) and better speed.

Acknowledgments
We would like to show our gratitude to the Research and Publishing Center of UIN Sunan Gunung Djati Bandung, for supporting publication of this article.

References
[1] Roman G and Davis R L 2001 Molecular biology and anatomy of Drosophila olfactory associative learning BioEssays 23(7) 571–581
[2] Netterstrøm I and Kayser L 2008 Learning to be a doctor while learning anatomy! Anat. Sci. Educ., 1(4) 154–158
[3] Alvarez S, Nikendei C and Schultz J H 2017 Development of a didactical training concept for peer tutors in gross anatomy Anat. Sci. Educ., 10(5) 495–502
[4] James H K, Chapman A W P, Dhukaram V, Wellings R and Abrahams P 2019 Learning anatomy of the foot and ankle using sagittal plastinates: A prospective randomized educational trial Foot 38 34–38
[5] Notebaert AJ 2009 Student perceptions about learning anatomy (Iowa City, Iowa, USA: University of Iowa)
[6] Lieu R M, Gutierrez A and Shaffer J F 2018 Student Perceived Difficulties in Learning Organ Systems in an Undergraduate Human Anatomy Course HAPS Educ., 22(1) 84–92
[7] Yosephine M and Prabowo Y D 2017 Pengembangan Aplikasi Pemeriksaan Kata Dasar dan Imbuhan pada Bahasa Indonesia Kalbiscentia 4(2) 118–130
[8] Nurachmandani S and Samsulhadi S 2010 Ilmu Pengetahuan Alam (Terpadu) Untuk SMP dan MTs kelas VIII (Jakarta: Pusat Perbukuan, Kementerian Pendidikan Nasional)
[9] Ferdinand F and Ariebowo M 2009 Buku Praktis Belajar Biologi 2 (IPA) Kelas 11 SMA (Jakarta: Pusat Perbukuan Departemen Pendidikan Nasional)
[10] Singla N and Garg D 2012 String matching algorithms and their applicability in various applications Int. J. Soft Comput. Eng. 1(6) 218–222