Jaro–Winkler Distance Improvement For Approximate String Search Using Indexing Data For Multiuser Application

F Friendly
Computer Engineering and Informatics Department, Politeknik Negeri Medan, Indonesia
friendly@polmed.ac.id

Abstract. Word searching method has been developed in many ways and named as: Hamming Distance, Jaccard Distance, Jaro Distance, Jaro-Winkler Distance, Levenshtein Distance, etc. Those methods are used for lexicographic comparison to find words according to the similarity of the words which searched. The time needed for searching by using these words distance method can cause overhead as some difference user might try to search the same words all over. If these method is used in a multi user application where the user generally searching for some keywords repeatedly, then the user might have a longer searching time compared to exact search. In spite of this problem, we try to propose a method where the first search result of the previous user, will be recorded to the database for future usage by indexing the search keywords. In order to try this method, we use Jaro-Winkler Distance method to search words. From the test result show that combining indexing and similarity word searching by using Jaro-Winkler Distance method can decrease the searching time to 90-92% compared to just using the Jaro-Winkler Distance method only. As the searched data increased, the processing time can be shorten.

1. Introduction

Searching for similarity words has been long found and developed by using many method that adapted from many statistic approached and fuzzy method. Among them are: Hamming Distance, Jaccard Distance, Jaro Distance, Jaro-Winkler Distance, Levenshtein Distance, Overlap Coefficient, Ratcliff-Obershelfp Similarity, Sorensen-Dice Distance, Tanimoto Coefficient etc. Those method are used in lexicographic comparison to find the close related words or similar word [1].

According to previous research [2], Jaro-Winkler method test shows a satisfactory result with score 0.87 in Mean Average Precision (MAP). These tests is done in searching for typewriting mistaken. The test result by using different method is shown in Figure 1.
Jaro-winkler Method generally used to find similarity among homogeneity data [3]. According to previous research, the effective and high rate in detecting words make this method a suitable candidate for testing this method. In this research [3], the effectiveness of Jaro-Winkler Method is shown in Table 1.

Table 1. Jaro-winkler Experiment Results [3]

| DBpedia Class | Size | OA(0.8) | OA(0.9) | OA(0.95) |
|---------------|------|---------|---------|----------|
| Actors        | 9509 | 15.07   | 10.13   | 6.38     |
| Architect     | 3544 | 5.58    | 5.58    | 2.32     |
| Criminal      | 5291 | 11.54   | 7.77    | 4.52     |

Table 1 shows by using a number of variant data by using variant size. Time needed to process each testing are large enough which is between 2,32-15,07 seconds. These data are used by a single user only and are not used by multiple users at once. When this method is used in a web based application, there will be overload to the server as the resources needed would be larger. Though this is not tested, but based on the time consume for executing this method show us this ineffectiveness. This ineffectiveness might grow higher as user search the same words all over again. Thus cause the time needed to process every request might be larger.

2. Approximate String Search Method

Approximate string search is a method which allow the search result is not the exact match to the keywords provide. Though the result were not the same, approximate string search is used to found mismatch in typing, or suggestion while the exact match were not found. Mistype is a common problem where the approximate string search method can be used as the solution.

This approximate string search is used by using similar word as the keyword. The similarity of the word that used can be described as:

1. Words with same or likely pronunciation
2. Words with the same or likely phonetic
3. Words with different allocation of character but with the same number of character
4. Words with the same number of character and have small number of different character
The following are the example of approximate string match: (a) *Kemarau and kemarin*, are words in Bahasa Indonesia language that have the same length (number of character) but some of the characters are different. By using approximate string match, these words are has similarity thus can be consider as equal. (b) *basa* and *basah* in Bahasa Indonesia is written in different character composition but have the same phonetic, thus these words are match.

Other than searching for similarity words, approximate string search are used in searching for the nucleotide order in a number of DNA data, thesaurus and spam filter in e-mail. The similarities of words to the keywords are measured by calculating the words exchanged by splitting and rearrange those words until the correct word is found. The result of this calculation is called distance.

The simple operation can be described as follow:

- Insertion: buh → buah
- Deletion: buah → bah
- Substitution: buah → buas
- Transposition: bias → bisa

These four simple operations can be re-generalized into three simple operations by adding NULL value as follow:

- Insertion: bu*h → buah
- Deletion: buah → b*ah
- Substitution: buah → buas

By using those character transposition operations, some word distance measurement method applied different value for each and every operation type.

### 2.1. Jaro-Winkler Method

This method is developed from Jaro Distance Metric, which is used to measure the similarity between 2 words. This method is used generally for duplicate words checker. The higher the Jaro Winkler distance between two words, than the similar those words are. The basic algorithm of this method is split into 3:

1. Counting the length of words
2. Find the similar character between these words.
3. Find the number of transposed character

To measure the distance/similarity rate \( d_j \) between words \( s_1 \) and \( s_2 \) by using the Jaro Distance method, the equation is used as follow:

\[
d_j = \left\{ \begin{array}{cl} 0 & \text{if } m = 0 \\ \frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) & \text{otherwise} \end{array} \right.
\]  

Where as :

\( m = \text{number of the same character with the same position} \)

\( |s_1| = \text{the length of the first word} \)

\( |s_2| = \text{the length of the second word} \)

\( t = \frac{1}{2} \text{ of the transpose character number} \)

This \( d_j \) value will become 0 when \( m=0 \). The parameter \( m \) is the number of character which is in the same position both in the first word and the second word where the distance is not more than 1 character. For example, these are the two words that is compared, *dekat* and *tekad*. The \( m \) value for both words is 3, which are for the character of \( e, k, a \). Though \( d \) and \( t \) character are in both words, but
the position of those characters are more than 1 character distance. The value of t is half the number of transpose characters. The words dekat and tekad each have 2 character that transpose which are t and d, thus the value of t=1. The calculation of dj for both words of dekat and tekad is as follow:

\[
dj = \frac{1}{3} \left( \frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c} \right) = \frac{1}{3} \left( \frac{3}{5} + \frac{3}{5} + \frac{3-1}{3} \right) = \frac{3}{3} = 0.622
\]

The words dekat and tekad can be said as similar when dj value is not more than the maximum value of the following equation.

\[
\text{Max value} = \left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1.
\]

For the word tekad and dekat, the maximum value is \((5/2)-1=1\). As dj value is 0.622 and not more than maximum value of 1, then both words have a similarity.

In Jaro-Winkler method, the distance between words (d_ω) further decide by the following equation:

\[
d_\omega = dj + (t\ell p(1 - dj))
\]

In Jaro-Winkler, prefix scale \((p)\) is a constant value which is given as an adjustment with value generally used is 0.1. The \(p\) value is used to give more adjustment and grading value in this equation. The \(t\) notation is the number of exact character at the beginning of each word. For the word tekad and dekat, \(t\) value is 0 while \(d_\omega\) value and \(dj\) value are equal.

3. Design Of Jaro Winkler Method By Using Indexing Data

3.1 Experiment Parameter and Observation

The parameter used in this experiment is name attribute of students. Student name is general information, which is commonly used by people and commonly search by people.

The measurement of the experiment is state as the time needed for the system to calculate the Jaro-Winkler Distance words in words searching used by multiple users at the same time. Access time thus compared between the direct implementation of Jaro-winkler method and the Jaro-Winkler method by using indexing. This result can show weather the propose method is better then the previous. The measurement will be round up by using Mean Average Precision (MAP). In this experiment, there will be 2 tables with 16.926 and 69.145 records. This is needed to analyze while using a large amount of data.

3.2 Experiment Model

Model that is used to make sure that the experiment is done in the right track. The methods are as follow:

- The model is arranged for data fetching and data management in the database.
- Jaro-winkler model is created to be used in both method testing.
- Jaro-winkler method and Jaro-winkler method by using data indexing will be tested by using some words.
The user access time that is used for approximate searching will be saved and analyzed.

### 3.3 Algorithm Design

In the process of saving the user access time, method of gathering data is needed. Though the result of approximate search by using Jaro-winkler is to be index, the first time a user search for un-index data, all the data must be compared by using this method. The result of comparison will be saved. Only the highest result of 20 will be saved. The rest of the data shall be ignored.

The information of number of records shall be saved as the approximate string search result by using this method can be differ when the data is added or remove. For the method algorithm, will be shown in Figure 2.

![Figure 2. Proposed Algorithm Design](image-url)
4. Method Testing

The testing used a number of determined names. The words used in this experiment shown in Table 2.

| Used Keywords             | ade  | nia  | ani  | ita  | adi  | rudi | mila | wati | andi | budi |
|---------------------------|------|------|------|------|------|------|------|------|------|------|
|                           | indah| benar| yonna| diana| anita| selena| vanesa| kirana| kuras| melati|
|                           | katerin| vanessa| permata| gunawan| kanelia| sulastr| sutrino| rolandha| benedith| meredith|
|                           | sumarjono| dukawati| kurniawati| hardiyanto| paragama| evangelina| diaph putri| kurniawati| hardiyanto| para digma|
|                           | sudharmanto| cokromino| bartolomeus| sri hartati| grande ansi|

These words are chosen randomly. The arrangement of the words is based on the length of the words. Each words length are: 3,4,5,6,7,8,9,10 and 11. For each search words we compare to 5 different words.

This experiment is held by using the 16,926 and 69,145 records data. Testing is done to find whether large data can influence the searching access time. The comparison methods used are as follow:

1. Jaro-Winkler searching method
2. Jaro-Winkler using indexing data search method

5. Result and Analysis

According to the experiment, the access time between the Jaro-Winkler and the Jaro-Winkler using indexing data shows no different at first search and shown the shortest access time for the next search of the same words. This is caused by the process of data indexing is done at the first time searching process.

Table 3 shows the result of searching method by using the number of length words. For Each length words and method is tested 20 times of searching repetition. Method 1 is state as the Jaro-Winkler method while method 2 is state as Jaro-Winkler using data indexing method. The measured value is in time unit of second.

| Word Length | Method 1 | Method 2 |
|-------------|----------|----------|
| 3 Words     | 6.942898 | 1.092073 |
| 4 Words     | 8.553997 | 1.152636 |
| 5 Words     | 11.44665 | 1.367764 |
| 6 Words     | 13.8234  | 2.01163  |
| 7 Words     | 16.96478 | 2.097179 |
| 8 Words     | 19.12848 | 2.737222 |
| 9 Words     | 21.44283 | 2.755292 |
| 10 Words    | 20.07115 | 2.898137 |
| 11 Words    | 19.41312 | 2.561623 |
Table 4. Access Time Improvement. This Table shows the decreased of access time used in both method.

| Number of Records | Improvement (times) | Improvement Percentage |
|-------------------|---------------------|------------------------|
| 16.926            | 1669.417            | 90.2033                |
| 69.145            | 1679.044            | 92.22565               |

Table 4 shown the improvement of access time 1669 times faster for 16.926 records and 1679 times faster for 69.145 records using Jaro-winkler using data indexing compare to only using the Jaro-winkler method.

By using MAP statistical counting, the average improvement of access time is 90% for data with 16.926 records and 92% for data with 69.145 records. The improvement is state as the decreased of access time. Thus shown the better improvement of Jaro-winkler method.

6. Conclusion

From the comparison of approximate searching method using Jaro-Winkler method and Jaro-winkler using data indexing method result can be conclude that by using Jaro-Winkler data indexing method, the access time decrease to 90-92 %, which shows the improvement of this method. The same measurement come up as similar in experiment for every first time search words for both method. Thus every first time search would result as the same access time, but for the next time search of the same keywords, the access time might reduce greatly.

Reference

[1] G. Navarro, "A Guided Tour to Approximate String Matching," ACM Computing Surveys, vol. 33, no. 1, pp. 31-88, 2001.
[2] Y. Rochmawati and R. Kusumaningrum, "Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks," Jurnal Buana Informatika, pp. 125-134, April 2016.
[3] K. Dressler and A.-C. N. Ngomo, "On the Efficient Execution of Bounded Jaro-Winkler Distances," 12 September 2014. [Online]. Available: http://www.semantic-web-journal.net/content/efficient-execution-bounded-jaro-winkler-distances. [Accessed 15 April 2017].
[4] P. E. Black, "Dictionary of Algorithms and Data Structures," National Institute of Standard and Technology, 27 May 2014. [Online]. Available: https://xlinux.nist.gov/dads/HTML/jaroWinkler.html. [Accessed 15 April 2017].
[5] I. T. Administration, "Damerau–Levenshtein distance usage,ITA's Data Services Platform," U.S. Exporting Data Authoritative, April 2017. [Online]. Available: http://developer.trade.gov/. [Accessed April 2017].
[6] L. BOYTSOV, "Indexing Methods for Approximate Dictionary Searching: Comparative Analysis," Association for Computing Machinery, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1963190.1963191. [Accessed 14 April 2017].
[7] W. E. Winkler, "Overview of Record Linkage," Statistical Research Division, U.S. Census Bureau, Washington, 2006.