International Journal of Research in Advent Technology, Vol.4, No.5, May 2016
E-ISSN: 2321-9637
Available online at www.ijrat.org

A Comparative Study on Name Matching Algorithms

Abha Chaudhary[1], Nidhi Wakchoure[2], Nilkamal Gotarne[3], Paulomi Nath[4]
Prof. Bhagyashree Dhakulkar[5]
Computer Engineering Department, Final Year Students DYPSOET, Pune, Maharashtra, India[1, 2, 3, 4]
Computer Engineering Department, Professor DYPSOET, Pune, Maharashtra, India[5]
Email: abha.bittu@gmail.com[1],*bhagyashree.dhakulkar@dypic.in*[5]

Abstract-Name matching plays an important role in many database and data mining applications. Data comparison and duplication detection helps in updating databases in organization as well as in identification purpose e.g. identifying individual using ID system. There are various matching techniques for comparison, record linkage, duplication detection etc. This paper gives basic description of several name matching algorithms which succeeded in dealing with name variation.

Index Terms- Name matching; Duplicate detection; Record linkage; Algorithms.

1. INTRODUCTION

Nowadays name matching plays an important in organizations as well as in identification purpose e.g. Identification of an individual using ID system, matching different datasets in an organization. This involves matching of string, for which many algorithms are proposed.

Many problems occur in searching and matching databases where it is important for a system to compare information or names of different people and to make a decision whether they are same or not. The matching algorithms are basically of three types. First type is based on sounds. List of rules are applied on the name or word to compress it into a sound code. Phonetic structure of a language is used in these algorithms in addition to standard sounds to match names which sound the same e.g. Soundex algorithm and its variants.

Another type is based on edit distance of two strings. To match two strings edits are used to make one string into another e.g. Monge-Elkan, Damerau-Levenshtein, Jaro-Winkler etc.

Third type is based on split strings into words or tokens. Algorithms fall into this category are Jaccard similarity and the TFIDF [1].String matching becomes problematic when variations and errors are more in names. Due to this, definite name matching leads to low results. It restricts identifying people, since it is not easy to find whether a name variation is a name of a different person or different spelling of the same person [2, 5].

The contribution of this paper is the outline of few of the matching algorithms.

2. NAME MATCHING ALGORITHMS

Following are few major name matching algorithms:

2.1. *Soundex*

Various regional names have range ethnic origins, that give us pronunciation of the names in same way but are spelled differently and vice versa e.g. “their” and “there”, both are pronounced same but spelled differently. There comes the soundex algorithm which is based on phonetic structure of language. In this, a written word is taken, such as individual’s name, as input, and character string is produced that determine a set of words which are phonetically alike. It uses a method which is based on phonetic classification of human speech sound. There are six phonetic classifications which are: Dental, Labiodentals, Velar, Alveolar, Glottal, and Bilabial. The vowels will not be encoded until it is the first letter while only consonants are encoded.

The algorithm involves following steps:

(1) Capitalize all letters and remove all punctuation.
(2) Keep the first letter of the word
(3) Convert all other occurrences of A, E, I, O, U, Y, H, W to 0.
(4) As per the following sets change letters into digit:
- B, F, P, V → 1
- C, G, J, K, Q, S, X, Z → 2
- D, T → 3
- L → 4
- M, N → 5
- R → 6

(5) All pair of digits that occur beside each other from the string resulted in step 3, remove it.

(6) Drop all zeroes from the string resulting in step 5 which are placed in step 3.
(7) Pad the resulted string with trailing zeroes and only return first four position, in the form of <FIRST LETTER><DIGIT><DIGIT><DIGIT>

For example: "Tiger" will be encoded as "T260" and "India" will be "I530"

2.2. *Damerau-Levenshtein*

This algorithm is based on edit distance between the strings. Edit distance is the minimum cost sequence of edits that converts one string to other i.e., source string to target string. The edits involve insertion, deletion, substitution, inclusion, transposition or reversing of character. Each edit is assigned with the cost and resultant cost is based on minimum number of edits made to convert source string to target.

As there are 5 allowed edits i.e. deletion, insertion, transposition of two adjacent character, substitution of single character and inclusion. One example of such: Distance between words Wait and Weight is 3

E.g.

Wait→ Weit (Substitution of 'a' to 'e')
Weit→ Weigt (Insertion of 'g')
Weigt→ Weight (Insertion of 'h')

2.3. *Gotoh-Smith-Waterman*

Gotoh-Smith-Waterman algorithm was developed to find matching substrings of DNA or protein. It is a dynamic matching technique which is similar to the edit distance. The main difference between both is that, smith-waterman allows gaps and even character specific match scores. It uses positive scores for matching and penalties for every mismatch and gap. A gap penalty is known as affine.

There are basically five operations:

- An accurate match between any two characters with score 5
- A fairly accurate match between any two alike characters
- A difference between any two characters with score -5
- Gap penalty with a score of -5
- Penalty on gap continuation with score -1

Space complexity is $O(|s_1| \times |s_2|)$ and time complexity is $O(\min(|s_1| \times |s_2|) \times |s_1| \times |s_2|)$.

2.4. *Monge-Elkan*

Monge-Elkan is a text string comparison method based upon internal character similarities measure in combination with token level similarities measure. A simple but very effective method for measuring the similarity between two text strings containing several tokens was proposed by Monge and Elkan. They used an internal similarity function $sim'(a,b)$ which is able to measure the similarity between individual tokens a and b. When given two texts A and B, |A| and |B| is their respective number of tokens, an inter-token measurement of similarity sim', monge-elkan computed as:

$$sim_{MongeElkan}(A,B) = \frac{1}{|A|}\sum_{i=1}^{|A|} \max_{j=1}^{|B|} \{sim'(a_i, b_j)\} \quad (1)$$

The time complexity of Monge-Elkan is $O(|A| \times |B|)$

2.5. *Jaro-Winkler*

The Jaro-Winkler is the measurement of similarities between given two strings. In the area of record linkage for duplicate detection this algorithm is mainly used. The similarity of the strings depends upon the jaro-winkler distance. If Higher the jaro-winkler distance then more similar is both the strings. This algorithm is most suitable for short strings such as proper names. Score is usually normalized in a way that 1 equates the exact match and 0 equates no similarity.

The distance d_j of given two strings s_1 and s_2:

$$d_j = \begin{cases} 0 & \text{If } m = 0 \\ \frac{1}{3}\left(\frac{n}{|s_1|} + \frac{n}{|s_2|} + \frac{n-t}{n}\right) & \text{otherwise} \end{cases}$$

Where,

n is number of matched characters.
t is of half the number of transpositions.

The two of the characters from s_1 and s_2 are count as matching only if they are the same and not farther than

$$\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1 .$$

Each character in s_1 is compared with its matching character in s_2. Number of matched character is divided by 2; it defines the number of transpositions.

E.g. CRATE comparing with TRACE, R, A, E are the matching characters that means n=3. Even if C and T appearing in both strings, they are farther than 1, here (5/2)-1=1. So, t=1.

Jaro-Winkler distance uses a prefix scale p which gives more favorable ratings to strings that match from the beginning for a set prefix length ℓ. Given two strings s_1 and s_2, their jaro-winkler distance d_w is:

$$d_w = d_j + \left(\ell_p \left(1 - d_j\right)\right)$$

Where:
d_j is the Jaro distance for strings s1 and s_2.
ℓ is the length of common prefix.
p is a constant scaling factor. It is for how much score is adjusted. Should not exceed 0.25, the distance can become larger otherwise.
Standard value of this constant in Winkler's work is p=0.1.

3. APPLICATIONS

- Employee verification: To verify job experiences, performance and other core details of new coming employee from other organizations.
- Identification system: Unique identification system, passport identification.
- Organizational purpose: For the analysis of organization's growth. E.g. matching customer record with survey record of company's customer to analyze profit or loss in revenue.

4. CONCLUSION

We have discussed some name matching algorithms which gives basic idea about string matching. There are three types of algorithm we studied. First type is on sounds of language, second on edit distance, and third on splitting string into tokens. Every algorithm works different each other and can be used in different fields on the basis of exact match as well as approximate match. We noticed that gotoh-smith-waterman works better for approximate match since it allows division of strings on the basis of gaps and Damerau-Levenshtein works better for exact match of short length strings.

Acknowledgments

We would like to thank Prof. Bhagyashree Dhakulkar for her guidance and immense dedication in providing us never ending knowledge for completing of domain research.

REFERENCES

[1] Philip Top, "A Dynamic Programming Algorithm for Name Matching" Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007).
[2] Peter Christen Department of Computer Science, the Australian National University, "A Comparison of Personal Name Matching: Techniques and Practical Issues", Sixth IEEE International Conference on Data Mining Workshops (ICDMW'06), 2006 IEEE.
[3] Chakkrit Snae, "A Comparison and Analysis of Name Matching Algorithms", International Scholarly and Scientific Research Innovation 1(1) 2007.
[4] Matteo Magnani, "A study on company name matching for database integration", Department of Computer Science, University of Bologna, Via Mura A.Zamboni 7, 40127 Bologna, Italy.
[5] Mikhail Bilenko and Raymond Mooney, University of Texas at Austin, "Adaptive Name Matching in Information Integration", IEEE INTELLIGENT SYSTEMS Published by the IEEE Computer Society.
[6] Timofey Medvedev, Alexander Ulanov, "Company Names Matching in the Large Patents Dataset", Copyright 2011 Hewlett-Packard Development Company, L.P.
[7] Truth Technologies, Inc., "Name and Address Matching Strategy", White Paper for Release December, 2010.
[8] Jeffrey Sukharev, "Parallel corpus approach for name matching in record linkage", 2014 IEEE International Conference on Data Mining.
[9] Antoon Bronselaer and Guy De Tr, "A Possibilistic Approach to String Comparison", IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL.17, NO. 1, FEBRUARY 2009.
[10] Sergio Jimenez, "Generalized Mongue-Elkan Method for Approximate Text String Comparison", A. Gelbukh (Ed.): Springer-Verlag Berlin Heidelberg 2009.
[11] M. Riedel, "High Productivity Data Processing Analytics Methods" with Applications MIPRO 2014, 26-30 May 2014, Opatija, Croatia.