The Review of Fields Similarity Estimation Methods

Mahsa Sabbagh Nobarian and Mohammad Reza Feizi Derakhshi

***Abstract*—Accuracy and consistency are the most important factors in any databases but increasing size of data has become a great challenge in this area. Detecting duplicate records is an important and very difficult process in huge databases containing millions of records. Field matching is a major process for duplicated record detection. In this paper, an attempt is made to provide a brief survey of field matching techniques and their efficiency.**

***Index Terms*—Duplicate detection, character based similarity metrics, edit distance, Jaro distance, Q-Grams.**

I. Introduction

With extending of the information technology, raw data plays an important role in organizations and is infrastructure of decision systems. Therefore increasing the accuracy of data will be imperative. This work will be complicated in massive database. On the other hand, integration of several databases with different representations of data reduces data accuracy.

Therefore existence of duplicate records is one of the causes that reduce data accuracy. Duplication occurs when database contains records that have different views of an identical entity. This variation of representation can be caused by integration of database with different structures or spelling and phonetic errors or abbreviations.

On the other hand, existence of duplicate records in database leads to negative impacts in process of knowledge acquisition from these resources. Therefore several techniques and algorithms have been presented for detection of duplicate records. Field matching technique is the first step in detection of duplicate records. The various algorithms have been represented for field similarity estimation and record matching step [1] [2] [3].Usage of data mining [4] in the area of data cleaning is as effective as in discovering reference data and validation rules from the data itself. Similarity metrics that are commonly used to detect similar field entries, and present an extensive set of duplicate detection algorithms presented in [5] that can detect approximately duplicate records in a database. The paper [6] presented two learnable text similarity measures suitable for estimation strings similarity: an extended variant of learnable string edit distance, and a novel vector-space based measure that employs a Support Vector Machine (SVM) for training.

According to data explosion phenomenon, duplicate detection in massive database will become imperative. For obtaining this goal, we describe the existence techniques for field matching phase in duplicate detection process then present performance of these approaches.

Manuscript received May 9, 2012; revised July 8, 2012.

Mohammad Reza Feizi Derakhshi is with Department of Computer, University of Tabriz, Tabriz, Iran (e-mail: mfeizi@tabrizu.ac.ir)

Mahsa Sabbagh Nobarian is with Department of Computer, Islamic Azad University, Shabestar Branch, Shabestar, Iran (e-mail: msn.sabbagh@yahoo.com)

II. Field Matching Methods

Field matching methods strongly depend on field type and content. So selecting the appropriate and effective fields will be important in duplication step. For example, the database containing people profiles, gender field can't effect on duplicate detection but name, family name and address fields can help to duplicate detection. On the other hand, more discussed fields in this process are from string type so in this paper string matching algorithms will be present.

III. Basic Concepts

If you are using *Word*, use either the Microsoft Equation Editor or the *MathType* add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation *or* MathType Equation). "Float over text" should *not* be selected.

IV. Units

In character based Matching algorithms, character is the smallest unit that is studied in field similarity estimation. These algorithms detect spelling or typing errors.
S1, S2 are strings and | s1 |, |s2 | show the strings length.
Existing operator to convert two strings together is as follows:

1) Insertion operation: insert a character into string and is shown with $\delta(\varepsilon, x)$.

2) Deletion operation: delete a character from string and is shown with $\delta(x, \varepsilon)$.

3) Substitution operation: replacing one character from s1 by a character from s2 and is shown with $\delta(a, b)$.

4) Transposition operation: transpose two characters in one string.

Div is a factor that can be calculated by one of three methods that are shown in (1),(2),(3):

$$\mathrm{div}_{sw} = \min(|s_1|, |s_2|) \quad (1)$$

$$\mathrm{div}_{sw} = \max(|s_1|, |s_2|) \quad (2)$$

$$\mathrm{div}_{sw} = \mathrm{avg}(|s_1|, |s_2|) \quad (3)$$

V. Edit Distance

The edit distance between two strings s_1 and s_2 is the Minimum number of edit operations of single characters needed to transform the string s_1 into s_2. There are insertion, deletion and Substitution operations. In the simplest form, each edit operation has cost 1. This version of edit distance is also referred to as the Levenshtein distance [5]. This technique gives the lowest cost of a sequence of operators. That is always smaller than or equal to length of longer field. To obtain this cost, distance matrix will be constructed for two strings. After obtaining the distance between two strings, amount of similarity will be calculated by using (4):

$$sim(s_1, s_2) = 1.0 - \frac{dist(s_1, s_2)}{div_{sw}} \tag{4}$$

The result of (4) will be between 0 and 1 that represents the amount of similarity between two fields.

VI. Edit Distance

This algorithm has only substitutions, which cost 1 in the simplified definition[7]. In the literature the search problem in many cases is called "string matching with *k* mismatches." The distance is symmetric, and it is finite whenever $|s_1|=|s_2|$. In this case it holds $0 \le d(s_1, s_2) \le |s_1|$.

VII. Episode Distance

This Algorithm has only insertions, which cost 1.In the literature the search problem in many cases is called "episode matching". This distance is not symmetric, and it may not be possible to convert s_1 into s_2 in this case [7].

VIII. Bag Distance

This algorithm has recently been proposed [8] as a cheap approximation to edit distance. A *bag* is defined as a multi set of the characters in a string for example, multi set ms('peter') = {'e', 'e', 'p', 'r', 't'}, and the bag distance between two strings is calculated as in (5) [9]:

$$\text{dist}_{\text{bag}}(s_1, s_2) = \max(|x - y|, |y - x|) \tag{5}$$

$x = ms(s_1)$, $y = ms(s_2)$ and $|\cdot|$ denoting the number of elements in a multi set[9]. For example: distbag ('peter', 'pedro') =distbag({'e', 'e', 'p', 'r', 't'}, {'d', 'e', 'o', 'p', 'r'}) = max(|{'e', 't'}|, |{'d', 'o'}|) = 2. Now the value of string similarity between strings is obtained by using (4).

IX. Smith-Waterman

This method is a dynamic programming algorithm. It was first developed to find optimal alignments between related DNA or protein sequences [5]. Five operations with particular weight exist in this algorithm:

- Weight of actual matching between the pairs of characters=2
- Weight of the approximate matching between pairs of characters (for example: m, n)=1
- Weight of mismatch between pairs of characters =-1
- Weight of gap in strings=-1

The Smith-Waterman algorithm has best score as main adjustable parameter. This parameter is calculated by construction matrix of match scores for each pair of symbols in the alphabet [10].The matrix should be constructed by using (6). W is assumed a cost of an operator and the values of H (i, 0) and H (0,j) can be obtained by (7):

$$H(i,j) = \max \begin{cases} 0 & \text{mismatch} \\ H(i-1, j-1) + W(a_i, b_j) & \text{match} \\ H(i-1, j) + W(a_i, -) & \text{delete} \\ H(i, j-1) + W(-, b_j) & \text{insert} \end{cases} \tag{6}$$

$$\begin{aligned} H(i, 0) &= 0 \quad , \ 0 \le i \le m \\ H(0, j) &= 0 \quad , \ 0 \le j \le n \end{aligned} \tag{7}$$

Bs_{sw} (*Best Score*) is the best and maximum value that is obtained by score matrix. div_{sw} is a factor that is obtained by (1),(2) or (3). Match score equal to two value that weight of matching characters. After obtaining these parameters, Strings Similarity can be calculated by using (8) as following:

$$\text{sim}_{sw}(s_1, s_2) = \frac{\text{bs}_{sw}}{\text{div}_{sw} \times \text{match score}} \tag{8}$$

For example: "peter" and "pedro" are two strings. In First step we should construct the score matrix from these strings by using (6). We show this matrix in Fig. 1:

	-	p	e	d	r	o	r	i	m	e	n
-	0	0	0	0	0	0	0	0	0	0	0
p	0	2	1	0	-1	-1	-1	-1	-1	-1	-1
e	0	1	4	3	2	1	0	-1	-2	1	0
t	0	0	3	3	2	1	0	-1	-2	0	0
e	0	-1	2	2	2	1	0	-1	-2	0	-1
r	0	-1	1	1	4	3	2	1	0	-1	-1
r	0	-1	0	0	3	3	5	4	3	2	1
i	0	-1	-1	-1	2	2	4	7	6	5	4
m	0	-1	-2	-2	1	1	3	6	9	8	7
a	0	-1	-2	-3	0	0	2	5	8	8	7
n	0	-1	-2	-3	-1	-1	1	4	7	7	**10**

bs_{sw}=10

Fig. 1. Score Matrix For Smith-Waterman

Now by using (8) and value of bs_{sw} , we can estimate strings similarity as follow:

$$\text{sim}_{sw}(s_1, s_2) = \frac{10}{11 \times 2} = 0.454$$

X. Damerau-Levenshtein Distance

In this variation of the Levenshtein distance a transposition is also considered to be an elementary edit operation with cost 1 [7] [11] (in the Levenshtein distance, a transposition corresponds to two edits: one insert and one delete or two substitutions). The sim_{dld} measure is calculated similarly to sim_{ld} [9]. "Fig. 2" shows an example:

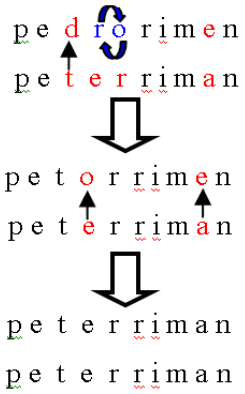

Fig. 2. Example of Damerau-Levenshtein Distance

The black and blue vectors in Fig. 2 show respectively Substitution and Transposition operations. According to "Fig.2" four operators exist for convert "pedro rimen" to "peter riman". Now strings similarity is estimated by using (4) and $\text{sim}(s_1, s_2) = 0.64$ is calculated.

XI. JARO Distance

Jaro [12] introduced a string comparison algorithm that was mainly used for comparison of last and first names. The basic algorithm for computing the Jaro metric for two strings s1, s2 is shown as in (9):

$$\text{Jaro} = \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c - t}{c} \right) \quad (9)$$

- c is the number of common characters between two strings.
- t is the number of transpositions; the number of transpositions is computed as follows: We compare the ith common character in s_1 with the ith common character in s_2. Each non matching character is a transposition [5] [13].

XII. JARO with Similar Characters

The string comparator program contains a list of pairs of characters that have been judged to be similar, so that they are more likely to be substituted for each other in misspelled words. After the common characters have been identified, the remaining characters of the strings are searched for similar pairs (within the search distance d). Each pair of similar characters increases the count of common characters by 0.3[14]. That is the similar character count is given by (10):

$$c_s = c + 0.3s \quad (10)$$

Then string similarity estimation is given by (11):

$$x_s = \frac{1}{3}\left(\frac{c_s}{|s_1|} + \frac{c_s}{|s_2|} + \frac{c_s - t}{c_s} \right) \quad (11)$$

Jaro with Common Prefix

This adjustment increases the score when the two strings have a common prefix. If p is the length of the common prefix, up to 4 characters, then the score x is adjusted to xp by (12) [14]:

$$x_p = jaro + \frac{p(1-x)}{10} \quad (12)$$

XIII. Longer String Adjustment

Finally there is one more adjustment in the default string comparator that adjusts for agreement between longer strings that have several common characters besides the above agreeing prefix characters [14]. The conditions for using the adjustment are:

- Both strings are at least 5 characters long.
- There are at least two common characters besides the agreeing prefix characters.
- We want the strings outside the common prefixes to be fairly rich in common characters, so that the remaining common characters are at least half the remaining common characters of the shorter string.

If all of these conditions are met, then length adjusted weight *xl* is computed by (13) [14]:

$$x_l = jaro + (1 - jaro).\frac{c - (p+1)}{|s_1| + |s_2| - 2(p-1)} \quad (13)$$

XIV. Q-Grams

Q-grams, also called n-grams [5][15] are sub-strings of length q from string s. For example, 'teacher' contains the bigrams 'te', 'ea', 'ac', 'ch', 'he' and 'er'. A q-gram similarity measure between two strings is calculated by counting the number of q-grams contained in both strings and divide by the average number of q-grams in both strings [9].

XV. Positional Q-Grams

A positional q-gram of a string s is a pair (i, s[i … i + q-1]), where s[i … i + q - 1] is the q-gram of s that starts at position i, counting on the extended string. The set G_s of all positional q-grams of a string s is the set of all the |s| + q - 1 pairs constructed from all q-grams of strings [15].On the other hands an extension to q-grams is to add positional information (location of a q-gram within a string) and to match only common q-grams that are within a maximum distance from each other. Positional q-grams can be padded with start and end characters similar to non-positional

q-grams, and similarity measures can be calculated in the same way as with non-positional q-grams [9].

XVI. Comparison of Field Matching Algorithms

In this paper, various techniques for filed matching were expressed. In this section these algorithms will be compared and concluded. First we reviewed the editing distance, smith-waterman, Jaro distance and q-grams algorithms in this paper. Edit distance algorithms appropriate for fast filtering for strings pair with great different length. But they are not appropriate for abbreviations and combination strings and lower accuracy of similarity estimation between them.

According to Smith-waterman, this approach uses weighted approach, when unmatching is between similar characters so obtaining upper amount of similarity for these strings. Other hands are assigned less weights for similar characters. So pairs of similar characters have less impact on mismatch. In addition to existing gaps in the fields also they have fewer impacts than any other operator. So it will be obtained upper similarity for abbreviations. But this method isn't appropriate for combination strings.

By considering that Jaro algorithms are based on the total number of common characters between two strings, so it is appropriate for shorter strings for example name and last name for people's profiles. But this method isn't appropriate for combination and abbreviations strings. Q-grams techniques are appropriate for abbreviations and combination strings. But q-grams construction will be needed to experts.

References

[1] A. Monge, "Matching Algorithms within a Duplicate Detection Syste," i*n proceedings of IEEE Data Engineering Bulletin,* 2000.

[2] Gu L. and R. Baxter R. "Adaptive Filtering for Efficient Record Linkage," *INSDM.* 2004.

[3] R. Duvall , R. kerber R and Thomas A. "Extending the Fellegi-Sunter Probabilistic Record Linkage Method for Approximate Field Comparator," *Biomedical Informatics in ElSEVIER.* 2010, pp.24-30.

[4] L. Ciszak , " Application of Clustering and Association Methods in Data Cleaning," *Computer Science and information Technology. IEEE.* 2008.

[5] A. Elmagadin, "duolicate record detection: a survey," *IEEE transactions on knowledge and data engineering.* 2007.

[6] M. Bilenko and R. Mooney ," Adaptive Duplicate Detection Using Learnable String Similarity Measures," *ACM SIGKDD Washington DC.*2003, PP:39-48.

[7] G. Navarro," AGuid Tour to Approximate String Matching," *ACM computing survay* .2001, pp. 31-88.

[8] I. Bartolini and P. Ciaccia, "Patella M. String matching with metric trees using an approximate distance," *In SPIRE, LNCS, Lisbon Portugal*, 2002; pp.271–283.

[9] P. Cheisten. P, "A Comparision of Personal Name Matching: Techniques and Practical Issues," *IEEE ICDM*, 2006, pp.290-294.

[10] A Monge and C Elkan, "The Field-Matching Problem:Algorithm and Applications," *ACM SIGKDD.* 1996, pp.267–270.

[11] F. Damerau, "A technique for computer detection and correction of spelling errors". *Communications of the ACM*, 1964, pp.171–176.

[12] M. Jaro and A. Unimatch, "A Record Linkage System: User's Manual," *technical report, US Bureau of the Census* Washington D.C.1976.

[1] W. Cohen, P. Ravikumar, and E. Fienberg, **"**A Comparison of String Metrics for Matching Names and Records". *American Association for Artificial Intelligence.* 2003.

[13] W. Yancey, "Evaluation Comparator Performance for Record Linkage," *TR in U.S census bureau.* 2005.

[14] L. Gravano , P. Peirotis, and H. Jagadish, "Using q-grams in a DBMS for Approximate String Processing," *IEEE Data Engineering Bulletin.*2 001, pp. 28-34.