2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

Presenting vector space based method in order to detect phonetic and spelling similarities

Mahsa Sabbagh-Nobarian [1]and Vahid Khalilpour [2]

[1] Young Researchers Club, Shabestar Branch, Islamic Azad University, Shabestar,Iran
[2] Islamic Azad University, Shabestar Branch, Iran

Abstract. According to this fact that nowadays raw data is considered as capital of organizations, increasing the extent of data accuracy will be inevitable. Duplicate detection is one of the steps that help more to increase the extent of data accuracy. In this article according to existence of phonetic errors in each language, one modern token based method in vector space and also jaro algorithm has been used according to one coding for letters of each language. Finally, the evaluation of purposed method was done by using jaro, jaro winkler techniques and inseparable string and typographic distance over real database that according to obtained results, the purposed algorithm showed 0.91 accuracy that it had accuracy increment compared with jaro, jaro winkler and inseparable string and typographic distance.

Keywords: Typographic Errors, Phonetic Matching, Field Matching

1. Introduction

Today by growing information technology, increasing data accuracy and their inductive information is essential. According to this fact that existence of phonetic and spelling errors in database causes decreasing of data accuracy, so the process of duplicate detection and determining similar cases in database will have significant role in the trend of increasing the extent of data accuracy.

Therefore, in this article first jaro, jaro winkler, inseparable string and typographic distance is presented that is one of the existing methods in the field of evaluating the similarity extent of fields. Then the purposed algorithm which is the combination of token based algorithms and character in one vector space is introduced and finally the purposed algorithm is evaluated by executing these algorithms on one real database related to people's profile.

2. The Process of duplicate detection

In most cases database has records that have different presentation from one identical entity. This variety in data presentation can be caused by integrating databases with different data structure but homological or due to existence of spelling, phonetic and using abbreviations. On the other hand, as stated existence of duplicate record will cause negative effects in processes of extracting information and acquiring knowledge from these references.

So detecting duplicate records is one of the essential steps in progress of increasing data accuracy. The first step of duplicate detection process is to estimate the similarity extent of fields. In the second step according to obtained values from estimating step of similarity extent, the similarity of whole record is studied.

According to this fact that existing errors are either typographic or phonetic, so in the following we will study some introduced algorithms in this field.

2.1. 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 [1]. 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 Eq. 1:

$$\mathrm{sim}(s_1, s_2) = 1.0 - \frac{\mathrm{dist}\ (s_1, s_2)}{\max(|s_1|, |s_2|)} \tag{1}$$

2.2. Jaro Distance

Jaro [2] 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 s_1, s_2 is shown as in Eq. 2 :

$$\mathrm{Jaro} = \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c - t}{c} \right) \tag{2}$$

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 [1] [3].

2.3. 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 Eq. 3 [4]:

$$x_p = jaro + \frac{p(1 - x)}{10} \tag{3}$$

2.4. Longer string Adjastment

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 [4]. 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 of the remaining common characters of the shorter string.

If all of these conditions are met, then length adjusted weight x_l is computed by Eq. 4 [4]:

$$x_l = jaro + (1 - jaro).\frac{c - (p + 1)}{|s_1| + |s_2| - 2(p - 1)} \tag{4}$$

2.5. Bag Distance

This algorithm has recently been proposed [5] as a cheap approximation to edit distance. A *bag* is defined as a multiset of the characters in a string for example, multiset ms('peter') = {'e', 'e', 'p', 'r', 't'}, and the bag distance between two strings is calculated as in Eq. 5 [6][7]:

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

$x = ms(s_1)$, $y = ms(s_2)$ and | · | denoting the number of elements in a multiset. For example:

$\mathrm{dist}_{\mathrm{bag}}$('peter', 'pedro') =$\mathrm{dist}_{\mathrm{bag}}$({'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 Eq. 4

2.6. Atomic String

Monge and Elkan [8] proposed a basic algorithm for matching text fields based on atomic strings. An atomic string is a sequence of alphanumeric characters delimited by punctuation characters. Two atomic strings match if they are equal or if one is the prefix of the other. Based on this algorithm, the similarity of two fields is the number of their matching atomic strings divided by their average number of atomic strings.

2.7. Token Based Purposed Method

Purposed algorithm estimates the similarity extent of two strings according to spelling and phonetic errors. According to this fact that in all the languages of world there are letters that have completely different spelling but have identical pronunciation, as a result while entering data, the existence of phonetic errors and also typographic errors are inevitable.

In purposed method first according to phonetic similarities of letters of one language it is tried to create one phonetic coding in letters of that language. For example, for English language we can use soundex coding method.

In this research in order to test purposed method, one real database that includes people's name and family name has been used.

Phonetic and spelling based purposed algorithm is presented as follows:

First two input strings are converted to a list of tokens. In fact, in one string it is the blank space between separator words of tokens.

Each token converts to corresponding phonetic code based on coding which is presented for one language at first like soundex coding.

A set of token aggregation of two strings are created then two vectors with length of the number of this set for two strings are created. In fact each component will be corresponded to one token.

The resulted codes from each token in the first string have been searched. If presented, that token has been removed from first string and related component of that token in the vector of first string receives the value of one. This process is done to the second string.

Each of the remained tokens from one string is compared with tokens of other string by using jaro technique that has been explained in previous section and their similarity is obtained according to Eq. 1. In fact, by using jaro technique over codes, the extent of similar characters based on phonetic and spelling is calculated for a pair of tokens.

At last the maximum value among obtained values is selected as value of selected token component from given string.

Now by having two correspondent vectors with two strings, we can obtain the similarity extent of two strings by using the angle between two vectors. The greater the angle the less the similarity of two strings and the lower the angle between two similar vector the greater the similarity of two strings. The cosine of angle between two numerical vectors is in a range of zero and one and it is calculated by using inner product of two vectors, so based on Eq. 6 we have:

$$\cos\alpha = \frac{v_1 . v_2}{|v_1| . |v_2|} \tag{6}$$

α is the angle between two vectors, v_1 and v_2 are correspondent vectors of first and second strings respectively, $|v_1|$ and $|v_2|$ are also the lengths of v_1 and v_2 vectors. By calculating the cosine of angle between two vectors, the more the obtained value reaches one, the more the extent of similarity.

3. Summary

In order to calculate the accuracy extent of purposed algorithm and its comparison with other above introduced algorithms, one real database that has the profile of people has been used that has the fields of name and family name. Since name and family name fields have more importance and has great influence of similarity detection of pair record compared with other fields, these two fields have been used in evaluation of algorithms.

In order to calculate accuracy the criteria of True Accept, True Reject, False Accept and False Reject have been used that in the following we will define each of them:

The True Accept criterion shows some cases that algorithm detected as similar cases truly that we show by TA.

The False Accept criterion shows some cases hat algorithm detected as similar cases incorrectly that we show by FA.

The True Reject shows some cases hat algorithm detected as dissimilar cases truly that we show by TR.

The False Reject shows some cases hat algorithm detected as dissimilar cases incorrectly that we show by FR.

The accuracy criterion can be calculated by using Eq. 7:

$$P = \frac{TR + TA}{TR + FR + TA + FA} \tag{7}$$

By executing introduced algorithms in article on a given database and by considering different thresholds, the values of criteria of True Accept, True Reject, False Accept and False Reject have been obtained for different executions of these algorithms based on Table 1:

Table 1: The obtained result by executing introduced algorithms

Edit Distance	Atomic String	Jaro Winkler	Jaro	purposed method	
0.5	0.4	0.8	0.7	0.65	threshold
87	87	87	87	87	True Accept
306	306	223	178	110	False Accept
864	832	915	960	1028	True Reject
0	0	0	0	0	False Reject
0.77	0.75	0.81	0.85	0.91	accuracy

According to obtained results, the token based purposed algorithm showed accuracy 0.91 that has accuracy increment of 0.06, 0.1, 0.16 and 0.14 in jaro, jaro winkler, Atomic string and typographic distance respectively.

4. References

[1] A. Elmagadin. Duplicate record detection: a survey. *IEEE Tansactions on Knowledge and Data Engineering*. 2007 19(1):1-16.

[2] M. A. Jaro. Unimatch. A Record Linkage System. User's Manual. *Technical Report US Bureau of the Census*, Washington D.C. 1976.

[3] W. Cohen, P. Ravikumar and E. Fienberg. A Comparison of String Metrics for Matching Names and Records, *Proc. ACM International Conference on Knowledge Discovery and Data Mining*. Washington, DC 2003: 8-24.

[4] L. Gravano, P. Peirotis and H. Jagadish. Using q-grams in a DBMS for Approximate String Processing. *IEEE Data Engineering Bulletin*, 2001, 24(4): 28-34.

[5] P. Ciaccia and M. Patella. String matching with metric trees using an approximate distance. *Lecture Notes in Computer Science*. Lisbon Portugal, 2002. 2476: 423–431.

[6] P. Cheisten. A Comparision of Personal Name Matching: Techniques and Practical Issues, *Proc. 6th IEEE International Conference on Mining Complex Data (ICDM 06)*, Hong Kong 2006: 12-20.

[7] G. Navarro. AGuided Tour to Approximate String Matching. *ACM computing survey*. 2001. 33(1):231-236.

[8] A. E. Mong and C. P. Elkan. The Field Matching Problem: Algorithms and Applications, *Proc. Second International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon. 1996. 2-8.