# Comparison and Improvement of Basic String Metrics for Surname Matching

Solmaz Khatami

Faculty of Engineering, Islamic Azad University of Sarvestan, Sarvestan, Iran

**Abstract:** For combining information from heterogeneous information sources, it's necessary to identify data records that refer to equivalent entities. Variations in representation across information sources can arise from differences in formats that store data, typological errors, abbreviations, and so on. This paper proposes three new techniques named token-based Jaro (TBJ), typological-error-based Jaro (TEBJ), and Jaro combining Soundex (JCS) for matching surnames to improve the field matching quality, and then compares these three algorithms with Jaro metric and Jaro-winkler metric –which are one of the basic descriptions of various field matching algorithms developed to find similarity metrics on the task of matching entity strings. We use a large real world database in Farsi which contains estate information. We utilize surname field of this database to examine and implement. There are typological errors more than other types of errors in our database, so we survey the mentioned algorithms on this kind of errors. According to conclusions; the precision of TEBJ is 0.95 and it is better in proportion Jaro that its precision is 0.88 and Jaro-winkler that its precision is 0.91 and TBJ that its precision is 0.93 and JCS which its precision is 0.92 in our dataset.
[Solmaz Khatami. **Comparison and Improvement of Basic String Metrics for Surname Matching.** *Life Sci J* 2013;10(5s):128-132] (ISSN:1097-8135). http://www.lifesciencesite.com. 23

**Keywords**:Jaro algorithm, Jaro-winkler algorithm, token-based Jaro (TBJ), typological error-based Jaro (TEBJ), precision, true negative, false negative

## I. INTRODUCTION

Large amounts of data are being created, communicated and stored by many individuals, organizations and business contains information about people. Even most scientific and technical documents contain details about their authors.

Finding and matching surnames is at the core of an increasing number of applications: from text and web mining, search engines, to information extraction, deduplication and data linkage systems. Variation and errors in surnames make exact string matching problematic, and approximate matching techniques have to be applied.

Matching of surname can be defined as the process of determining whether two surname strings are instances of the same surname [5]. As surname variations and errors are quite common [6], exact name string comparison will not result in good matching quality. Rather, an approximate measure of how similarity measure between 1.0 (two names are identical) and 0.0 (two names are totally different) is used.

The two main approaches for matching surnames are phonetic encoding and pattern matching. Many techniques have been developed for both approaches, and several techniques combine the two with the aim to improve the matching quality.

The contributions of this paper are an overview of two most commonly used techniques for matching surnames: Jaro and Jaro-winkler similarity metrics, proposing three new techniques for matching surnames, and a comparison of their performance using a large real world data set containing surnames.

## II. JARO DISTANCE METRIC

Jaro [1] 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 $s_1$ and $s_2$ includes the following steps:

*1) Compute the string lengths $|s_1|$ and $|s_2|$,*

*2) Find the "common characters" c in the two strings; common are all the characters $s_1[i]$ and $s_2[j]$ for which $s_1[i] = s_2[j]$ and $|i-j| \leq \frac{1}{2}\min\{|s_1|, |s_2|\}$.*

*3) Find the number of transpositions t; the number of transpositions is computed as follows: we compute the ith common character in $s_1$ with the ith common character in $s_2$. Each non-matching character is a transposition.*

The Jaro comparison value is:

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

From the description of the Jaro algorithm, we can see that the Jaro algorithm requires O ($|s_1|.|s_2|$) time for two strings of length $|s_1|$ and $|s_2|$, mainly due to the step 2 that computes the "common characters" in the two strings.

## III. JARO-WINKLER DISTANCE METRIC

The Jaro-winkler distance is a measure of similarity between two strings. It is a variant of the Jaro distance metric and mainly used in the area of record linkage (duplicate detection). Winkler and

Thibaudeau [2] modified the Jaro metric to give higher weight to prefix matches, since prefix matches are generally more important for surname matching. The score is normalized such that 0 equates to no similarity and 1 is an exact match. 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:

$$d_w = d_j + (lp(1 - d_j)) \quad (2)$$

Where:

- $l$ is the length of common prefix at the start of the string up to a maximum of 4 characters,
- $p$ is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. P should not exceed 0.25, otherwise the distance can become larger than 1. The standard value for this constant in Winkler's work is p=0.1,
- $d_j$ is the Jaro distance for strings $s_1$ and $s_2$.

## IV. TOKEN-BASED JARO

Since we use a real world database for comparisons and this database has special data, we decided to do improvement base on these data.

Names in Farsi are similar when prefix of them or suffix of them is similar. Even in some situations that name created from many portions such as "George William John Smith" if the middle of names in two strings are alike it causes to increment similarity measure. In such data, being equal of characters does not guaranty the similarity. For example "William" and "Winkler" are two completely different surnames although they have three same characters: "w", "i", and "l". Jaro distance metric gives high similarity measure to these different strings that is 0.71, but it is not logical answer. Because of such examples exist in our database; we decide to do some changes in Jaro and Jaro-winkler distances to give improve precision on our data.

Data in surname field in our database created from many portions. So In this algorithm we uses token instead of character. We consider each word as a token. Then compute the number of similar tokens and put it in c. $m_1$ is the number of tokens in first string and $m_2$ is the number of token in second string. Transposition (t) in this algorithm is for some surnames like "George William" and "William George" that creates from two tokens: "George" and "William". First token in first string is second token in second string and second token in first string is first token in second string.

$$sim(s_1, s_2) = \frac{1}{3}\left(\frac{c}{m_1} + \frac{c}{m_2} + \frac{c-t}{c}\right) \quad (3)$$

In above example c=2, t=1, $m_1$=2, $m_2$=2; So,

sim ($s_1$, $s_2$) =0.83

Whereas;

Jaro ($s_1$, $s_2$) = 0.58, Jaro-wink ($s_1$, $s_2$) =0.58.

## V. TIPOLOGICAL-ERROR-BASED JARO

When we're typing a letter or a document, we may type wrong words. There are a lot of reasons to type wrong words. One of these reasons is that we maybe on a hurry and because some of letters is next to each other on keyboard, it increments the percent of these mistakes. For example "M" and "N" are next to each other on keyboard. Also "o" and "p" are one another example. It is possible that we change angle of our fingers when we're typing and this change is unconscious. Also we may substitute one letter for another such as "to" and "ot". We may do not write a letter such as "word" and "wod".

A lot of these kinds of errors are seen in our dataset. With considering this probability; we display equal the letters which are next to each other. Also we consider the probability of mistakes. It means that because Farsi is the right to left language, the probability of typing letters that are the right of correct letter is more than other directions. We use the letter that has highest probability for displaying equal with correct letter. We do this because each word on keyboard has eight directions and we could not consider nine letters equal. t is the number of transpositions .

$$sim(s_1, s_2) = \frac{1}{2}\left(\frac{c}{\max(|s_1|, |s_2|)} + \frac{c-t}{c}\right) \quad (4)$$

$$c = m_1 + m_2 * w \quad (5)$$

$m_1$ is the number of common characters, $m_2$ is the number of characters which are different in two strings. w is the weight of characters that we have placed them in $m_2$. To compare two strings if there is just one different character w is 0.9. Increasing different characters makes the w lower. It means that if there are two different characters w is 0.8. if there are three different characters it gets 0.7 and so on. Consider $s_1$=Appolonia college, $s_2$=Appolonia colege:

$m_1$=16, $m_2$=1.

c= 16+1*0.9=16.9, sim ($s_1$, $s_2$) = 0.997.

whereas;

Jaro ($s_1$, $s_2$) = 0.98, Jaro-wink ($s_1$, $s_2$) = 1.006.

We draw your attention to another example:

$s_1$=Appolonia college, $s_2$=Apoloni colege:

c=14+3*0.7=0.97

Whereas;

Jaro ($s_1$, $s_2$) = 0.94, Jaro-wink ($s_1$, $s_2$) = 0.952.

It is clear that the result of Jaro-winkler algorithm is Irrational. The conclusion is typological-error-based algorithm works better than jaro and jaro-winkler algorithms in these kinds of errors.

## VI. JARO COMBINING SOUNDEX

### A. Soundex

Strings may be phonetically similar even if they are not similar in a character or token level. For example the word "Kageonne" is phonetically similar to "Cajun" despite the fact that the string representations are very different. The phonetic similarity metrics are trying to address such issues and match such strings.

Soundex, invented by Russell [3], [4] is the most common phonetic coding scheme. Soundex is based on the assignment of identical code digits to phonetically similar groups of consonants and is used mainly to match surnames.

In Farsi the rules of Soundex coding are as follows:

*1) Assign the following codes to the alphabetic letters of Farsi:*

*a) ع, ع, ا = 1 (as a in English)*
*b) ف, و, پ, ب, ب = 2(as d, b, p, f in English)*
*c) ط, ت = 3 (as t in English)*
*d) ص, س, ث = 4 (as s in English)*
*e) ژ, ش = 5 (as sh, zh in English)*
*f) ج, چ = 6 (as j, ch in English)*
*g) ح, ه = 7 (as h in English)*
*h) خ = 8 (as kh in English)*
*i) ظ, ض, ز, ذ = 9 (as z in English)*
*j) ر, ی =10 (as r, y, i in English)*
*k) گ, غ, ق =11 (as g, gh in English)*
*l) ک = 12 (as k in English)*
*m) م, ن = 13 (as m, n in english)*
*n) ل =14 (as l in English)*

*2) Vowels are not coded but serve as separators;*

*3) Drop the separators;*

*4) In comparison of two strings if one of their letters are fewer than another pad with zeroes.*

### B. Jaro combinig Soundex

In Jaro algorithm instead of putting matched characters in c; we put the number of matching codes in c and drop the third division of formula, because of that the sentence that is inside of parentheses has divide 2. $m_1$ is the number of codes in first string and $m_2$ is the number of codes in second string.

$$sim(s_1, s_2) = \frac{1}{2}\left(\frac{c}{m_1} + \frac{c}{m_2}\right) \qquad (5)$$

For example matching codes for "علی" (as Ali in English) are 1, 14, and 10.

## VII. IMPLEMENTATION & EVALUATION

We have implemented these four algorithms in c#.net programming language. We have used real database for examinations and implementations. Database includes estate information.

We utilize surname field of database. This field contains 106 strings. We have 105 comparisons for each string and totally 11130 comparisons for whole of database which is computed in this way: 106*105=11130. We gain 11130 similarity measures with these comparisons for each algorithm.

Then we have to compute precision, false negative, and true negative of each algorithm to identify that which algorithm work well for surname in Farsi on our database.

For computing precision of algorithm we have to use human intelligent to identify real similarity. It means that when I look at this paper I know that "William" and "William" are similar strings. But "William" and "Wilam" are not equal. However, system could do mistakes. We count the number of 1's that algorithm has gained. Then count the number of 1's that we ourselves have gained them. The 1's that system and us have gained together plus 0's that system and us have gained divided to the number of comparisons (11130) is precision of algorithm.

True negative means that the result of algorithm is correct whereas real result is incorrect. False negative means that the result of algorithm is incorrect whereas real result is correct.

For identifying these three metrics we define a threshold that begins from 0.4 up to 0.95. For example if threshold is 0.6, we consider the numbers which are greater than 0.6, correct; and the numbers which are less than 0.6 are incorrect.

In consequence we try to select the threshold for algorithms that its precision is highest and its false negative is lowest.

We gained the results for Jaro distance as in (1), Jaro-winkler distance as in (2), TBJ as in (3), TEBJ as in (4), and JCS as in (5).

For Jaro distance:
Precision is 0.88,
True negative is 0.12,
False negative is 0.
For Jaro-winkler distance:
Precision is 0.91,
True negative is 0.09,
False negative is 0.
For Jaro based on token:
Precision is 0.93
True negative is 0.07,
False negative is 0.
For Jaro based on typological error:
Precision is 0.95,
True negative is 0.05,
False negative is 0.
For Jaro combining Soundex:
Precision is 0.92
True negative is 0.08,
False negative is 0.

## CONCLUSIONS

Improving previous algorithms on special dataset is a fundamental task which we tried to reach it in

this paper. We use real dataset based on estate information and utilize the surname field of this database to do examinations. Some strings of this field has made from many portions so we try to use tokens instead of characters in our algorithm. Some strings contain typological error so we tried to improve Jaro algorithm based on these kinds of errors. And some of them contain phonetic similarity. Then we compare the gained results of four algorithms: Jaro distance metric, Jaro-winkler distance metric, and the improved algorithms named token-based Jaro, typological-error-based Jaro, and Jaro combining Soundex. The result of comparison is three final numbers to each algorithm. These numbers refer to precision, false negative, and true negative. The best result is the result that the precision is highest and false negative is lowest.

TEBJ gives the best result with precision: 0.95 and false negative: 0. For Jaro, Jaro-winkler, TBJ, and JCS; precisions in sequence are 0.88, 0.91, 0.93, and 0.92.

TABLE I. THE RESULT OF COMPARISON FOR JARO

| False negative | True negative | Precision | Threshold |
|---|---|---|---|
| 0 | 0.13 | 0.77 | 0.4 |
| 0 | 0.2 | 0.8 | 0.45 |
| 0 | 0.18 | 0.82 | 0.5 |
| 0 | 0.15 | 0.85 | 0.55 |
| 0 | 0.13 | 0.87 | 0.6 |
| 0 | 0.12 | 0.88 | 0.65 |
| 0.03 | 0.09 | 0.88 | 0.7 |
| 0.09 | 0.06 | 0.85 | 0.75 |
| 0.18 | 0.03 | 0.79 | 0.8 |
| 0.24 | 0.02 | 0.74 | 0.85 |
| 0.26 | 0.0009 | 0.73 | 0.9 |
| 0.34 | 0.0006 | 0.65 | 0.95 |

TABLE II. THE RESULT OF COMPARISON FOR JARO-WINK

| False negative | True negative | Precision | Threshold |
|---|---|---|---|
| 0 | 0.17 | 0.83 | 0.4 |
| 0 | 0.16 | 0.84 | 0.45 |
| 0 | 0.15 | 0.85 | 0.5 |
| 0 | 0.12 | 0.88 | 0.55 |
| 0 | 0.1 | 0.9 | 0.6 |
| 0 | 0.09 | 0.91 | 0.65 |
| 0.03 | 0.03 | 0.94 | 0.7 |
| 0.07 | 0.02 | 0.91 | 0.75 |
| 0.12 | 0.01 | 0.87 | 0.8 |
| 0.16 | 0.0008 | 0.83 | 0.85 |
| 0.20 | 0.0007 | 0.79 | 0.9 |
| 0.28 | 0.0005 | 0.71 | 0.95 |

TABLE III. THE RESULT OF COMPARISON FOR TBJ

| False negative | True negative | Precision | Threshold |
|---|---|---|---|
| 0 | 0.17 | 0.83 | 0.4 |
| 0 | 0.16 | 0.84 | 0.45 |
| 0 | 0.14 | 0.86 | 0.5 |
| 0 | 0.11 | 0.89 | 0.55 |
| 0 | 0.09 | 0.91 | 0.6 |
| 0 | 0.08 | 0.92 | 0.65 |
| 0 | 0.07 | 0.93 | 0.7 |
| 0.06 | 0.04 | 0.9 | 0.75 |
| 0.16 | 0.01 | 0.83 | 0.8 |
| 0.25 | 0.009 | 0.74 | 0.85 |
| 0.3 | 0.004 | 0.69 | 0.9 |
| 0.33 | 0.0001 | 0.66 | 0.95 |

TABLE IV. THE RESULT OF COMPARISON FOR TEBJ

| False negative | True negative | Precision | Threshold |
|---|---|---|---|
| 0 | 0.17 | 0.83 | 0.4 |
| 0 | 0.16 | 0.84 | 0.45 |
| 0 | 0.13 | 0.87 | 0.5 |
| 0 | 0.09 | 0.91 | 0.55 |
| 0 | 0.08 | 0.92 | 0.6 |
| 0 | 0.07 | 0.93 | 0.65 |
| 0 | 0.05 | 0.95 | 0.7 |
| 0.1 | 0.02 | 0.88 | 0.75 |
| 0.14 | 0.008 | 0.85 | 0.8 |
| 0.20 | 0.001 | 0.79 | 0.85 |
| 0.27 | 0.0006 | 0.72 | 0.9 |
| 0.31 | 0.00009 | 0.68 | 0.95 |

TABLE V. THE RESULT OF COMPARISON FOR JCS

| False negative | True negative | Precision | Threshold |
|---|---|---|---|
| 0 | 0.2 | 0.8 | 0.4 |
| 0 | 0.16 | 0.84 | 0.45 |
| 0 | 0.14 | 0.86 | 0.5 |
| 0 | 0.09 | 0.91 | 0.55 |
| 0 | 0.08 | 0.92 | 0.6 |
| 0.03 | 0.07 | 0.9 | 0.65 |
| 0.1 | 0.05 | 0.85 | 0.7 |
| 0.15 | 0.02 | 0.83 | 0.75 |
| 0.24 | 0.002 | 0.75 | 0.8 |
| 0.26 | 0.001 | 0.73 | 0.85 |
| 0.29 | 0.0006 | 0.7 | 0.9 |
| 0.35 | 0.00009 | 0.64 | 0.95 |

REFERENCES

[1] Gonzalo Navarro. *"A guided tour to approximate string matching."* ACM Computing Surveys, 33(1):31-88, 2001.

[2] Solmaz Khatami and Mohamad-Reza Feizi-Derakhshi. "Comparison and Improvement of Basic String Metrics for Surname Matching." Submitted to IASBS 2011.

[3] Pinheiro and Sun. *"A program for extracting probable matches from a large file for record linkage."* Technical Report Statistical Research Report Series RRC2002/01, U.S. Bureau of Censes, Washington, D.C., March 2002.

[4] William E. Winkler. Methods for record linkage and baysian networks. Technical Report Statistical Research Report Series RRS2008/05, US. Bureau of the Cencus, Washington, D.C. 2008.

[5] Sudipto Guha, Nick Koudas, Amit Marathe, and Divesh Srivastava. Merging the results of approximate match operations. In proceedings of the 30th International Conference on Very Large Databases (VLDB 2004), pages 636-647,2004.

[6] Eugene Agichtein and Venkatesh Ganti. Mining reference tables for automatic text segmentation. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2006), pages 20-29, 2006.

[7] Howard B. Newcombe, James M.Kennedy, S.J. Axford, and A.P. James. Automatic linkage of vital records. Science, 130(3381):954.959, October 2009.

2/24/2013