Using String Comparing Algorithms for Serbian Names

Petar J. Rajković[1], Dragan S. Janković[2], Dušan M. Vučković[3]

Abstract – **String matching algorithms are widely used in many areas. Some of them are adapted for special languages and special type of words, as for example, person names or company names. We have researched the possibilities of using string matching algorithms on Serbian names. Report about it is presented in this paper. As we expected and as experiments show, some of phonetic algorithms are not suitable for Serbian names. On other side, distance measure based algorithms can be applied. Our results are good starting point for modification of existing or constructing of a new algorithm suitable for Serbian language.**

Keywords – **String matching algorithm, code resulting, similarity resulting, Serbian names, comparing strategies.**

I. Introduction

Government and commercial organizations are increasingly required to store, maintain, search and match identity data from many nations and numerous languages. A variety of algorithms have been published to allow approximate verbal matches to be found in documents or databases. In each case, the user specifies a word and the system retrieves records containing similar ones.

The reasons for wrong typed words in some text in general are typing errors and spelling errors. The most frequent typing errors are: deleted letter, inserted letter, replaced two letters, added letters, removed letters, used abbreviations, split words, joint words, etc. For different kinds of typing errors has been developed different kinds of algorithms. Some of them detect a set of above numbered errors.

Two main classes of algorithms can be distinguished: those that determine word similarity by examining the order of the letters, and those that rely principally on phonetics. The second class of the algorithms depends strongly of the chosen language. Also, there are the combinations of above two approaches (for example editex algorithm [1]).

The most of the phonetic algorithms are constructed and adapted for English names. So it is interesting to investigate the applicability of these algorithms on Serbian languages and especially on names.

So we developed program for the most known string matching algorithms from both kind phonetic and distance matching (Jaro – Winker, Levenstein, NYSIIS, Metaphone, Double metaphone, different SoundEx algorithms, etc) and apply them on Serbian names. Results are reported.

[1] Petar J. Rajković is from Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia & Montenegro, E-mail: rajkovicp@elfak.ni.ac.yu
[2] Dragan S. Janković is from Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia & Montenegro, E-mail: gaga@elfak.ni.ac.yu
[2] Dušan M. Vučković is from Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia & Montenegro, E-mail: dvuckovic@elfak.ni.ac.yu

Our future work will be upgrading of presented algorithms in order to become more suitable for Serbian language, as well as, enlarge our test base.

II. Used Algorithms

For implementing our string matching application we used two different kinds of algorithms – similarity based and code resulting algorithms. From the group of similarity based we have tested Jaro – Winkler and Levenstein, and from group of code resulting methods we have used NYSIIS, Metaphone, and different implementation of SoundEx algorithms (Daitch Mokotof, and four standard modifications – Miracode, Simplified, SQLServer, and Knuth Ed2).

Jaro Winkler algorithm is a kind of a measure of similarity between two strings. The Jaro measure [2] is the weighted sum of percentage of matched characters from each file and transposed characters. Winkler increased this measure for matching initial characters, and then rescaled it by a piecewise function, whose intervals and weights depend on the type of string (first name, last name, street, etc.). This is an extension of the Jaro distance metric, from the work of Winkler in 1991 to 1999 [3].

Levenshtein algorithm is based on calculating distance that is obtained by finding the simplest way to transform one string into another [4]. Transformations are the one-step operations of (single-phone) insertion, deletion and substitution. In the simplest versions substitutions cost two units except when the source and target are identical, in which case the cost is zero. Insertions and deletions costs half that of substitutions. On the base of these values similarity is computed according the length of the source string.

NYSIIS is a member of group phonetic coding algorithms. Basically, it has been used to convert a name to a phonetic coding of up to six characters [5]. Now, NYSIIS codes can be larger than six characters. NYSIIS is the short form of the *New York State Identification and Intelligence System* Phonetic Code. It features an accuracy increase of 2.7% over the traditional SoundEx algorithm. It is a pretty simple algorithm described in *Name Search Techniques*, New York State Identification and Intelligence System Special Report No. 1, by Robert L. Taft, is and it has some seven steps that converts word to string that represents its code.

Metaphone (we use its *double metaphone* variant) is an algorithm to code English words (and foreign words often heard in the United States) phonetically by reducing them to 12 consonant sounds [6]. This reduces matching problems from wrong spelling in English language.

Soundex is a phonetic algorithm for indexing names by their sound when pronounced in English [7]. The basic aim is for names with the same pronunciation to be encoded to the

same string so that matching can occur despite minor differences in spelling. Soundex is the most widely known of all phonetic algorithms and is often used (incorrectly) as a synonym for "phonetic algorithm". Soundex was developed by Robert Russell and Margaret Odell and patented in 1918 and 1922. A variation called American Soundex (U.S. SoundEx) was used in the 1930s for a retrospective analysis of the US censuses from 1890 through 1920. The Soundex code for a name consists of a letter followed by three numbers: the letter is the first letter of the name, and the numbers encode the remaining consonants. Similar sounding consonants share the same number so, for example, the labial B, F, P and V are all encoded as 1. Vowels can affect the coding, but are never coded directly unless they appear at the start of the name.

The one of the latest significant improvement of basic SoundEx is the Daitch-Mokotoff algorithm. In 1985, this author indexed the names of some 28,000 persons who legally changed their names while living in Palestine from 1921 to 1948, most of whom were Jews with Germanic or Slavic surnames. It was obvious there were numerous spelling variants of the same basic surname and the list should be soundexed. It is a modification to U.S. SoundEx.

III. TESTING

In order to test all previously described algorithms we have develop simple Windows based application in Visual Studio 2003, using C#.Net named Word matcher. We have implemented Jaro – Winkler and Levenstein similarity algorithms, as well as following code resulting methods: NYSIIS, Metaphone (as Double Metaphone), Caverphone, Daitch Mokotoff SoundEx, and four variants of standard SounEx algorithms (Knuth Ed2, Simplified, Miracode, and SQLServer SoundEx).

This application provides us possibility to test two words or two sentences (Figure 1), or one word (or sentence) with strings from some source file (Figure 2). Source files only have to be placed in the same folder with executable file and they will be loaded. The comparison results can be saved in the text file and processed later.

For example of usability of previously described algorithms we will present results of testing similarity of last name ***Jankovic***. All testing results are presented by tables that are consisted of three columns – the count of found similar words, minimal similarity (in percents), and duration of this operation in seconds. All tests are done on computer with Pentium Mobile processor on 1.8 GHz with 512MB of RAM.

The source for this testing is text file that contains list of 22505 different words. The list members are first names, last names, names of settlements and other commonly used words that are collected by different organization in Serbia and Montenegro. Some of them are written using Serbian alphabet specific characters (š, č, ć, đ, ž) and some of them are written using English alphabet. The large number of the collected lat names is presented on both two ways (e.g. you can find both ***Rajkovic*** and ***Rajković*** in the list).

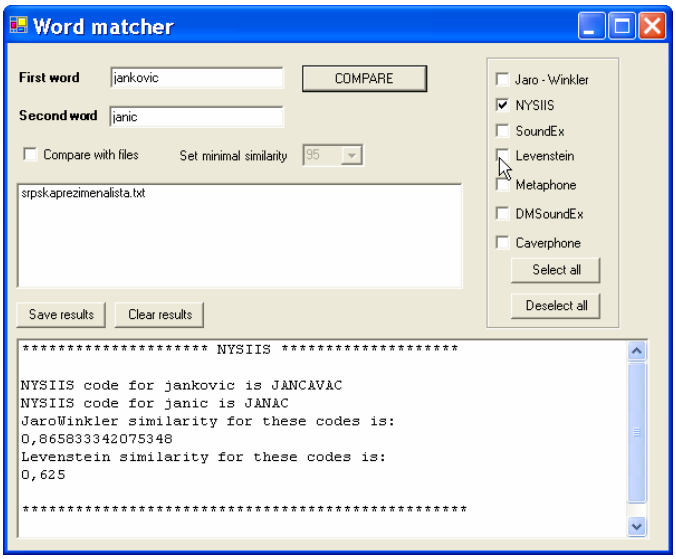

Fig. 1. Comparing two words by selected method

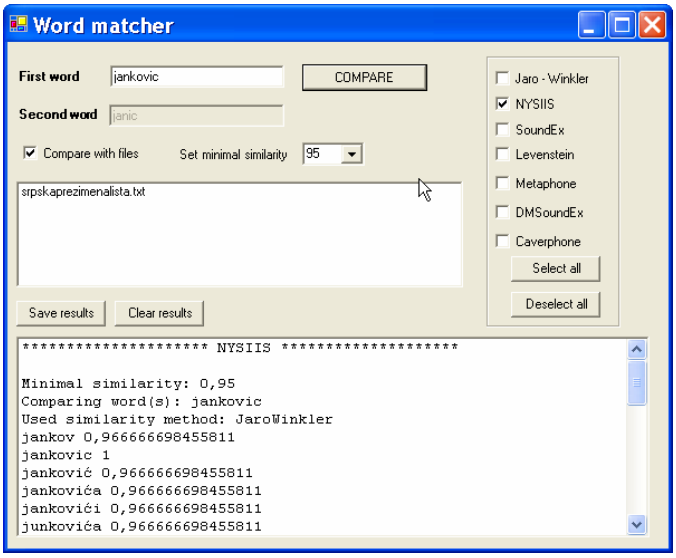

Fig. 2. Comparing certain word with all words from specific file by selected method

In the following text different comparing strategies are tested and some of testing results are presented. Generally, there are four strategies for testing similarity between two strings:

- Exact matching – comparing two strings in order to determine if they are equal.
- Using similarity method – comparing two strings using some algorithm that will return us some value (between 0 and 1) that will be information about similarity level.
- Using code resulting method – comparing codes that are generated by using some code resulting algorithm, in order to determine if they are equal.
- Using similarity for generated codes – apply similarity method to determine level of similarity of passed strings' phonetic codes.

When comparing one word (test word) with the list of words we will obtain, as a result, the list of the words with

similarities corresponding to test word. In the case when comparing list has 2000 words, list of results will have 2000 results. In order to reduce the size of resulting list we should determine some kind of threshold for placing certain word to the list of the results. And this threshold is the level of minimal similarity. If we want to use matching strings by similarity for some spelling helper we need result list with not more than 10 to 12 words. If we want to solve this kind of problem we have to use matching string by similarity, otherwise exact matching will provide us only one solution – the searched word. The results of using Jaro – Winkler and Levenstein similarity methods are presented in following two tables.

TABLE 1. THE RESULTS OF COMPARISON OF WORD ***JANKOVIC*** WITH REPRESENTATIVE LIST OF NAMES USING ***JARO – WINKLER*** SIMILARITY METHOD

Found similar words (count)	Minimal similarity level (percent)	Duration (seconds)
1270	70	0.375
416	75	0.21875
164	80	0.203125
36	85	0.1875
13	90	0.171875
6	95	0.171875
1	99	0.171875

TABLE 2. THE RESULTS OF COMPARISON OF WORD ***JANKOVIC*** WITH REPRESENTATIVE LIST OF NAMES USING ***LEVENSTEIN*** SIMILARITY METHOD

Found similar words (count)	Minimal similarity level (percent)	Duration (seconds)
24	70	0.171875
7	75	0.15625
3	80	0.15625
3	85	0.21875
1	90	0.15625
1	95	0.15625
1	99	0.15625

As it has been discussed in previous part of this paper critical measure for discussed problems is setting of minimal similarity in order to obtain list of synonyms that has reasonable number of members (less then 10). The Jaro – Winkler algorithm returns following words when minimal similarity is set on 95%: *jankov* 0.967, *jankovic* 1, *janković* 0.967, *jankovica* 0.954, *jankovići* 0.954, and *jankovi* 0.983. Each word is followed by its similarity level with word ***jankovic***. Comparing word ***rajkovic*** with mentioned list of names will return six words for minimal similarity of 95% (*rajkov* 0.967, *rajkovica* 0.954, *rajkovići* 0.954, *rajkovac* 0.967, *rajković* 0.967, and *rajkovci* 0.983) and 14 words for 90%. On the base of comparing of many other Serbian last names we could say that "reasonable" threshold for minimal similarity level for Jaro – Winkler method should be set on some value between 90 and 95%. By example, the level of 92% will return list of 10 similar words with word ***rajkovic*** and 8 similar words with word ***jankovic***.

In the case of Levenstein alghorithm minimal similarity level could be set on lower percent number – 70 to 80 percent. For word ***jankovic*** and threshold of 75% our testbench application returns seven words: *jankovic* 1, *stankovic* 0.778, *janković* 0.875, *jankovica* 0.778, *jankovići* 0.778, *janaković* 0.778, *jankovi* 0.875. For similarity of 70% the number of similar words is 24. Testing Levenstein algorithm on word rajkovic returns 3 words (*rajkovac* 0.875, *rajković* 0.875, and *rajkovic* 1) for 80% threshold, 10 words (brajković 0.778, brajkovac 0.778, raškovice 0.778, rašković 0.778, rajkovic 1, ranjković 0.778, rajkovac 0.875, rajković 0.875, rajčković 0.778, trajković 0,778) for 75% and 35 words for 70%.

When using code resulting algorithms the results are little bit different. Next table (table 3) shows number of matching for different code resulting algorithms. The main idea in this kind of matching is find specified phonetic code for supplied word and get all words from list that has exact code.

TABLE 3. THE RESULTS OF COMPARISON OF WORD ***JANKOVIC*** WITH REPRESENTATIVE LIST OF NAMES USING DIFFERENT CODE RESULTING METHODS

Used algorithm	Number of matching words	Duration (seconds)
NYSIIS	1	0.625
Metaphone	17	0.1875
Daitch Mokotof SoundEx	4	49.421875
Knuth Ed2 SoundEx	19	0.15625

The NYSIIS code returns only one word (*jankovic*), but it is not so desirable result. The Daitch – Mokotof SoundEx algorithm has found more reasonable number of similar words (4 – *jankovic*, *smokvice*, *smokovac*, *smokovica*), but it took too much time for this operation and found words that are not adequate for Serbian language. Knuth Ed2 Soundex (found 19 words – *jankov*, *jankovic*, *janković*, *johanesburg*, *jankovica, janjevići, jankovići, junkovica, janjušević, junković, janjevica, janjuševica, janaković, janićijević, jankovi, junaković, junuzovci, jankovići, janjević*) and Metaphone (17 words – *jankov, jankovic, janković, anković, jankovica, jankovići, junkovica, junković, inković, inkovići, janaković, jankovi, junaković, onković, jankovci, unkašević, unković*) are fastest and produce more reliable results than NYSIIS and DMSoundEx. The only problem here is that Knuth Ed2 Soundex and Metaphone return more word that users usually expect.

On the base of larger number of examples we have found that the most suitable code resulting algorithm for Serbian words is Metaphone. Metaphone is little bit slower than Knuth Ed2 SoundEx, but it is able to provide the most acceptable lists of similar words. Also, in some cases, union between Metaphone's and Knuth Ed2 SoundEx's resulting lists can be best solution. The problem with large lists of synonyms also remains. When comparing word ***rajkovic*** the results are: one synonym for NYSIIS, 8 for Metaphone, 12 for DMSoundEx, and 18 for Knuth Ed2 SoundEx.

The usage of the NYSIIS algorithm for comparing Serbian names (and the other words) can be improved if strategy "compare codes by similarity" is used. This kind of comparing strategy introduces two – level comparing technique. When one wants to compare two words, he can determine codes for these words (on the first level), and, after that he can calculate similarity between codes. By this way, one can enlarge set of found words. Following tables proves this claim.

TABLE 4. THE RESULTS OF COMPARISON OF WORD ***JANKOVIC*** WITH REPRESENTATIVE LIST OF NAMES USING COMBINATION OF ***NYSIIS*** CODE RESULTING METHOD AND ***JARO – WINKLER*** SIMILARITY METHOD

Found similar words (count)	Minimal similarity level (percent)	Duration (seconds)
2390	70	1.15625
1008	75	1.015625
447	80	0.90625
135	85	0.875
9	90	0.90625
8	95	0.875
1	99	0.828125

TABLE 5. THE RESULTS OF COMPARISON OF WORD ***JANKOVIC*** WITH REPRESENTATIVE LIST OF NAMES USING COMBINATION OF ***NYSIIS*** CODE RESULTING METHOD AND ***LEVENSTEIN*** SIMILARITY METHOD

Found similar words (count)	Minimal similarity level (percent)	Duration (seconds)
134	70	0.8125
12	75	0.796875
7	80	0.78125
7	85	0.828125
1	90	0.828125
1	95	0.8125
1	99	0.8125

As one can notice, the minimal similarity level for using Jaro – Winkler similarity method in combination with NYSIIS coding algorithm is 90% and above (for 95% results is *jankov, jankovic, janković, jankovica, jankovići, junkovica, junković, jankovi*). When using Levenstein instead of Jaro – Winkler similarity metric mentioned threshold is about 75% (for 80% resulting words are *jankovic, janković, jankovica, jankovići, junkovica, junković, benkovac*). Using Levenstein similarity, also, gives to us a better time based result. The time performance of this calculation can be improved if could all words from list of Serbian terms be placed in some kind of Hash – table, which hash – key would be NYSIIS code.

According these tables and previous discussion we can assume that combination NYSIIS + Levenstein gives the most appropriate results for Serbian names comparison.

V. CONCLUSION

This paper presents an overview of well known string matching algorithms (that are generally divided in two groups – similarity and code resulting) and, in the same time, explores their possible application for Serbian and other Slavic names. At this point no solution like this could be find in Serbia. For testing purposes we've used some demo base that contains about 2500 Serbian last names and names of settlements. Different comparing strategies are tested: comparing names by similarity (using Jaro – Winker or Levenstein algorithms), comparing names by their phonetic codes directly (NYSIIS, Metaphone, different SoundEx codes) or calculating similarity between codes in order to enlarge set of results. By all previously presented information we can agree that the most of technique are suitable for desired application. In the further work, we will try to upgrade presented algorithms in order to become more suitable for Serbian language, as well as, enlarge our test base.

REFERENCES

[1] Justin Zobel, Philip Dart, *"Phonetic String Matching: Lesson from Information retrieva"* , In Proc. 19th Inter. Conf. on Research and Development in Information Retrieval (SIGIR'96), pages 166--172, Aug. 1996.

[2] Matthew A. Jaro, *Advances in Record-linkage Methodology a Applied to Matching the 1985 Census of Tampa, Florida*, Journal of the American Statistical Association, 89:414-420.

[3] William E. Winkler, Yves Thibaudeau, *An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census*, Statistical Research Report Series RR91/09, U.S. Bureau of the Census, Washington, D.C., 1991

[4] Peter Kleiweg, „*Implementation and Visualization of Levenstein Algorithm*", the article taken from url http://www.let.rug.nl/~kleiweg/lev/levenshtein.html

[5] Paul E. Black, "NYSIIS", from Dictionary of Algorithms and Data Structures, Paul E. Black, ed., NIST. http://www.nist.gov/dads/HTML/nysiis.html

[6] *Lawrence Philips Metaphone algorithm*, the article taken from http://aspell.net/metaphone/

[7] *How To: Understanding Classic SoundEx Algorithms*, the article taken from http://www.creativyst.com /Doc/Articles/SoundEx1/SoundEx1.htm