ISSN 0005-1179, Automation and Remote Control, 2020, Vol. 81, No. X, pp. 345–359. © Pleiades Publishing, Ltd., 2020.
Original Russian Text © V.S. Vykhovanets, J. Du, S.A. Sakulin, 2018, published in Upravlenie Bol'shimi Sistemami, 2018, No. 73, pp. 67–94.

LARGE-SCALE SYSTEMS CONTROL

An Overview of Phonetic Encoding Algorithms

V. S. Vykhovanets*, J. Du, and S. A. Sakulin****

*Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
**Bauman Moscow State Technical University, Moscow, Russia
e-mail: valery@vykhovanets.ru, forsola@qq.com, ss141291@yandex.ru

Received September 12, 2017
Revised ???
Accepted ???

AbstractThis paper presents an overview of the phonetic encoding algorithms designed to determine the similarity of words in sound (pronunciation). Phonetic encoding algorithms are divided into the algorithms for comparing words and the algorithms for determining the distance between words. Word comparison algorithms, such as SoundEx, NYSIIS, Daitch–Mokotoff, Metaphone, and Polyphone, as well as algorithms for determining the distance between words, such as Levenshtein, Jaro, and N-grams, are described. For each algorithm, the advantages and shortcomings are discussed, and an analog for the Russian language is given. For eliminating the common shortcomings of phonetic encoding algorithms, the idea suggested in this paper is to use not the letter sequences of words, but the sequences of their elementary sounds. In this case, word recognition, record linkage, and word indexing by sounds are expected to improve.

Keywords: phonetic encoding algorithm, phonetic distance, record linkage, word indexing by sound

1. INTRODUCTION

Phonetic encoding algorithms are algorithms for word indexing by sound, which use a letter sequence of a word and pronunciation rules to convert them into text (code, index, or key). If the encoding texts for two different words coincide or are close to one another, the resulting conclusion is that these words are similar in sound.

The first phonetic encoding algorithm for the English language, SoundEx [69], was used in the 1930s to encode surnames in a population census. The algorithm is based on an encoding method designed to eliminate spelling and typographical errors in names. For example, SoundEx will calculate the same encoding text "S530" for such words as "Smith," "Smithe," and "Smyth," i.e., will identify these words as identical in sound.

With the development of computer technologies, many other phonetic encoding algorithms appeared, including the ones for other natural languages. Examples of such algorithms are NYSIIS [77], the advanced SoundEx [75], Metaphone [40], and others.

Phonetic encoding algorithms include not only algorithms for comparing words, but also algorithms for determining the distance between words in the case of searching by sound. The algorithms for calculating the Levenshtein distance [2] and the Jaro distance [30], as well as the distance based on N-grams [33] are most widespread in applications.

Phonetic encoding algorithms are widely used in the fields where the comparison of acoustic data with text samples is required. Some example include speech recognition, word spelling check and correction, database search, extraction and mining, user identification, cross-language speech encoding, Internet search, etc.

Despite the emergence and development of new speech recognition models and methods, such as probabilistic and statistical algorithms, hidden Markov models, neural networks, machine learning, etc., phonetic encoding algorithms have not lost their relevance, since they are basic for using these models and methods in practice [14, 22, 44].

This paper gives an overview of the phonetic encoding algorithms. The main goal is to consider the solutions intended for comparing strings by sound, as well as to identify recent trends in their development.

2. BASIC ALGORITHMS

2.1. SoundEx

This algorithm was patented almost a hundred years ago [69]. SoundEx for the English language consists of five steps as follows.

Step 1. Retain the first letter of the word.

Step 2. Delete the letters A, E, I, O, U, Y, W, and H from the word.

Step 3. Replace the remaining letters by the digits according to Table 1.

Table 1. Letter encoding in SoundEx

Digit	Letters
1	B, P, F, V
2	C, S, K, G, J, Q, X, Z
3	D, T
4	L
5	M, N
6	R

Step 4. If the code contains a group of identical digits, replace this group by the first digit, with the exception of the digits separating the letters W and H in the original word.

Step 5. Generate the resulting code from the first letter of the word and the first three digits obtained in the previous steps. If the code has less than three digits, augment the code with zeros.

Example 1. Consider the following two words: “Lee” and “Shaw.” The algorithm described above will yield the encoding text “L000” for the word “Lee” and “S000” for the word “Shaw.”

Example 2. To explain Step 4, consider the word “Ashcraft.” Due to the presence of Step 4, the encoding text for this word will be “A226,” not “A261.”

As Table 1 shows, the basic letter encoding principle of the algorithm is that the letters close in sound are encoded by the same digit, and the unpronounceable letters are deleted. However, the algorithm has several shortcomings as follows.

First, there are words similar in sound that have different encoding texts.

Example 3. For the words “Lee” and “Leigh” having the same sound, we get different codes, “L000” and “L200,” respectively.

The second shortcoming is opposite to the first one: there are words different in sound that have the same encoding text.

Example 4. The words “Gauss” and “Ghosh” with different sounds have the code “G200.”

To partially eliminate the problems revealed during operation, a modified version of the SoundEx algorithm with the following encoding table is currently used for the English language.

Table 2. Letters encoding

Digit	Letters
1	B, P
2	F, V
3	C, S, K
4	G, J
5	Q, X, Z
6	D, T
7	L
8	M, N
9	R

Clearly, new groups with their own digital codes are formed from groups 1 and 2 of the original SoundEx algorithm. In addition, the length of the encoding text in the modified SoundEx algorithm is not limited to four symbols.

In accordance with experimental evidence, up to 21 surnames have the same value of the SoundEx code. In the modified algorithm, this indicator does not exceed 2 or 3 surnames.

SoundEx is strongly language-dependent. Therefore, many modifications of this algorithm were developed for various languages such as Chinese [20], Spanish [7], Persian [36], Arabic [56], Malay [53], etc. For the Russian language, the letter encoding is in accordance with Table 3, and the letters to be deleted are *У, Е, Ё, Ы, А, О, Э, Я, И, Ю, Ь,* and *Ъ*.

Table 3. Encoding of Russian letters in SoundEx

Digit	Letters
1	*Б, П*
2	*Ф, В*
3	*Ж, З, С, Х*
4	*К, Г*
5	*Ц, Ч, Ш, Щ*
6	*Д, Т*
7	*Л, Й*
8	*М, Н*
9	*Р*

Despite its shortcomings, the SoundEx algorithm has positive reviews [37]. Currently, the algorithm is widely used to compare words by sound and significantly increases the probability of identifying words. Due to simplicity and low computational complexity, this algorithm has become standard and is integrated into the search engine of almost all known database management systems [27].

2.2. NYSIIS

This algorithm was developed in 1970 for the New York State Identification and Intelligence System [77]. NYSIIS yields slightly better results than SoundEx, owing to more complex rules for converting the source word to the code.

The algorithm takes into account the pronunciation of English words and consists of six steps; see below.

Step 1. Convert the word prefix (the beginning of the word) using the following substitutions: $MAC \to MCC$; $KN \to N$; $K \to C$; $PH, PF \to FF$; $SCH \to SSS$.

Step 2. Convert the word suffix (the end of the word) using the following substitutions: $EE \to Y$; $IE \to Y$; $DT, RT, RD, NT, ND \to D$.

Step 3. Convert the entire word using the following substitutions: $EV \to AF$; $A, E, I, O, U \to A$; $Q \to G$; $Z \to S$; $M \to N$; $KN \to N$; $K \to C$; $SCH \to SSS$; $PH \to FF$; $W \to A$.

Step 4. Delete H after all vowels, and delete S and A at the end of the word.

Step 5. Convert the word suffix using the substitution $AY \to Y$.

Step 6. Restrict the resulting code to 6 symbols only.

As it follows from the description above, NYSIIS uses a large number of rules relating spelling with pronunciation. Also, this algorithm takes into account the functioning of vowels when pronouncing words: all vowels are replaced by vowel A.

A comparative study of the phonetic encoding algorithms demonstrated that NYSIIS is most applicable for encoding surnames, where it gives excellent results.

Example 5. The surnames "Brain," "Brown," and "Brun" have the code "Bran"; the surnames "Capp," "Cope," "Copp," and "Kipp," the code "Cap"; the surnames "Dane," "Dean," "Dent," and "Dionne," the code"Dan"; the surnames "Smith," "Schmit," and "Schmidt" have the code "Snat"; the surnames "Trueman" and "Truman," the code "Tranan" [27].

2.3. Daitch–Mokotoff SoundEx

In the course of using phonetic algorithms, it was established that SoundEx and NYSIIS poorly manage the words of other languages. For taking into account the specifics of word pronunciation in other languages, the Daitch–Mokotoff SoundEx algorithm was developed, after the names of the authors. It has several improvements, such as a longer code length and a proper consideration of different word pronunciations, expressed in the multiple encoding of the same word.

This algorithm includes more complex rules for converting an original word into its code. Like in NYSIIS, not only single letters, but also their sequences are involved in the formation of the resulting code. A word is converted into a numerical code according to the following table (see [75]).

Table 4. Letter encoding in improved SoundEx algorithm

Letter sequence	B	V	O
$AI, AJ, AY, EI, EY, EJ, OI, OJ, OY, UI, UJ, UY$	0	1	
AU	0	7	
IA, IE, IO, IU	1		
EU	1	1	
A, UE, E, I, O, U, Y	0		
J	1	1	1
$SCHTSCH, SCHTSH, SCHTCH, SHTCH, SHCH, SHTSH,$ $STCH, STSCH, STRZ, STRS, STSH, SZCZ, SZCS$	2	4	4
$SHT, SCHT, SCHD, ST, SZT, SHD, SZD, SD$	2	43	43
$CSZ, CZS, CS, CZ, DRZ, DRS, DSH, DS, DZH, DZS, DZ, TRZ,$ $TRS, TRCH, TSH, TTSZ, TTZ, TZS, TSZ, SZ, TTCH, TCH, TTSCH,$ $ZSCH, ZHSH, SCH, SH, TTS, TC, TS, TZ, ZH, ZS$	4	4	4
SC	2	4	4
DT, D, TH, T	3	3	3
CHS, KS, X	5	54	54
S, Z	4	4	4
CH, CK, C, G, KH, K, Q	5	5	5
MN, NM	66	66	66
M, N	6	6	6
$FB, B, PH, PF, F, P, V, W$	7	7	7
H	5	5	
L	8	8	8
R	9	9	9

The order of conversions corresponds to the order of the letter sequences in Table 4. Here the columns B, V, and O specify the digital codes for the letter sequence from the first column as follows: B, at the beginning of a word; V, before a vowel; O, in other cases.

Alternative word codes with different pronunciations are obtained for the words formed from an original word using the following substitutions: $CH \to KH, TCH$; $CK \to K, TSK$; $C \to K, TZ$; $J \to Y, DZH$; $RS \to RTZ, ZH$.

Example 6. For the surname "Peters" with the SoundEx code "P362," the Daitch–Mokotoff SoundEx algorithm will yield two codes, namely, "739400" for the form "Peters" and "734000" for the form "Petertz." For the surname "Jackson" with the SoundEx code "J250," the Daitch–Mokotoff SoundEx algorithm will yield even four codes, namely, "154600," "454600," "145460," and "445460" for the forms "Jackson," "Yakson," "Jakson," and "Jatskon," respectively.

2.4. Metaphone

This is another phonetic encoding algorithm for words, with due consideration of the basic rules of the English language, which was developed in 1990; see [40]. It differs from the previously mentioned algorithms in more complex conversion rules. Also, letters are not divided into groups and are not encoded by numbers. Metaphone outputs a variable-length code consisting of letters.

This algorithm includes 16 steps as follows [10].

Step 1. Delete the repeated neighbor letters, except for the letter C.

Step 2. Convert the word prefix using the following substitutions: $KN \to N$; $GN \to N$; $PN \to N$; $AE \to E$; $WR \to R$.

Step 3. Delete the word suffix MB.

Step 4. Convert a letter sequence with the letter C using the following substitutions: $CIA \to XIA$; $SCH \to SKH$; $CH \to XH$; $CI \to SI$; $CE \to SE$; $CY \to SY$; $C \to K$.

Step 5. Convert a letter sequence with the letter D using the following substitutions: $DGE \to JGE$; $DGY \to JGY$; $DGI \to JGY$; $D \to T$.

Step 6. Perform the substitution $GH \to H$ if GH is not at the end of the word and not before a vowel.

Step 7. Convert the word suffix using the substitutions $GN \to N$ and $GNED \to NED$.

Step 8. Convert a letter sequence with the letter G using the following substitutions: $GI \to JI$; $GE \to JE$; $GY \to JY$; $G \to K$.

Step 9. Delete the letter H after vowels, but not before vowels.

Step 10. Convert the entire word using the following substitutions: $CK \to K$; $PH \to F$; $Q \to K$; $V \to F$; $Z \to S$.

Step 11. Convert a letter sequence with the letter S using the following substitutions: $SH \to XH$; $SIO \to XIO$; $SIA \to XIA$.

Step 12. Convert a letter sequence with the letter T using the following substitutions: $TIA \to XIA$; $TIO \to XIO$; $TH \to 0$; $TCH \to CH$.

Step 13. Convert the word prefix using the substitution $WH \to W$. If there is no vowel after the letter W, delete the latter.

Step 14. Convert the word prefix using the substitution $X \to S$; in the middle of the word, substitute $X \to KS$.

Step 15. Delete the letters Y that are not before vowels.

Step 16. Delete all vowels, except for the initial one.

Example 7. The surnames "Brain," "Brown," and "Brun" have the code "BRN"; the surnames "Capp," "Cope," "Copp," and "Kipp," the code"KP"; the surnames "Dane," "Dean," and

"Dionne," the code "TN." At the same time, the surname "Dent" has the code "TNT" (see Example 5); the surname "Smith," the code "SM0"; the surname"Schmit," the code "SXMT"; the surname "Schmidt," the code "SXMTT"; the surnames "Trueman" and "Truman," the code "TRMN."

In 2000, a second version of the algorithm was developed, the so-called Double Metaphone [41]. Unlike the original version applicable to the English language only, Double Metaphone takes into account the pronunciation of words borrowed from other languages [41]. For these words, the algorithm yields two codes, one for each pronunciation.

Although Double Metaphone has advantages over Metaphone, it still has some limitations [45]. In particular, words with different pronunciations and the same code still occur: for example, the words "Alice," "Elsa," and "Ullos" are encoded as "ALS."

In 2009, the third commercial version of this algorithm, Metaphone 3, appeared. As it was declared [58], the new algorithm increases the accuracy of word identification from 89% (Double Metaphone) to 98 % (Metaphone 3). In addition, Metaphone 3 began supporting the borrowed words from more languages. This algorithm is rather complex and includes a large number of rules. Its description in the Java language takes more than seven thousand lines [42].

Metaphone was adapted to the Russian language [1]. For the Russian language, this algorithm consists of five steps as follows.

Step 1. Convert vowels using the following substitutions: *О, Ы, Я* $\to$ *А*; *Ю* $\to$ *У*; *Е, Ё, Э, ЙО, ЙЕ* $\to$ *И*.

Step 2. Devocalize the consonant letters that are followed by any consonant, except for *Л, М, Н,* and *Р*, or consonants at the end of the word using the following substitutions: *Б* $\to$ *П*; *З* $\to$ *С*; *Д* $\to$ *Т*; *В* $\to$ *Ф*; *Г* $\to$ *К*.

Step 3. Delete the repeated letters.

Step 4. Convert the word suffix using the following substitutions: *УК, ЮК* $\to 0$; *ИНА* $\to 1$; *ИК, ЕК* $\to 2$; *НКО* $\to 3$; *ОВ, ЕВ, ИЕВ, ЕЕВ* $\to 4$; *ЫХ, ИХ* $\to 5$; *АЯ* $\to 6$; *ЫЙ, ИЙ* $\to 7$; *ИН* $\to 8$; *ОВА, ЕВА, ИЕВА, ЕЕВА* $\to 9$; *ОВСКИЙ* $\to$ @; *ЕВСКИЙ* $\to$ #; *ОВСКАЯ* $\to$ \$; *ЕВСКАЯ* $\to$ %.

Step 5. Delete of the letters *Ъ, Ь* and the hyphen.

However, due to the small number of rules, Metaphone for the Russian language does not identify some phonetically similar words.

2.5. Polyphone

A special phonetic encoding algorithm, Polyphone, was developed for the Russian language; see [59]. In the course of the development procedure, the morphological, phonological, phonetic, and historical aspects of word pronunciation in the Russian language were taken into consideration.

This algorithm consists of the following steps.

Step 1. Convert Latin letters to the Cyrillic letters of the same form using the following substitutions: $A \to A$; $E \to E$; $O \to O$; $C \to C$; $X \to X$; $B \to B$; $M \to M$; $H \to H$. The last three substitutions are used only for the capital letters.

Step 2. Delete all letters that do not belong to the Cyrillic alphabet.

Step 3. Delete the letters *Ь, Ъ*.

Step 4. Replace two similar letters by the one.

Step 5. Replace the single letters using the following rules: *А, Е, Ё, И, О, Ы, Э, Я* $\to$ *А*; *Б* $\to$ *П*; *В* $\to$ *Ф*; *Г* $\to$ *К*; *Д* $\to$ *Т*; *З* $\to$ *С*; *Щ* $\to$ *Ш*; *Ж* $\to$ *Ш*; *М* $\to$ *Н*; *Ю* $\to$ *У*.

Step 6. Convert the entire word using the following substitutions: *АКА* → *АФА*; *АН* → *Н*; *ЗЧ* → *Ш*; *ЛНЦ* → *НЦ*; *ЛФСТФ* → *ЛСТФ*; *НАТ* → *Н*; *НТЦ* → *НЦ*; *НТ* → *Н*; *НТА* → *НА*; *НТК* → *НК*; *НТС* → *НС*; *НТСК* → *НСК*; *НТШ* → *НШ*; *ОКО* → *ОФО*; *ПАЛ* → *ПЛ*; *РТЧ* → *РЧ*; *РТЦ* → *РЦ*; *СП* → *СФ*; *ТСЯ* → *Ц*; *СТЛ* → *СЛ*; *СТН* → *СН*; *СЧ* → *Ш*; *СШ* → *Ш*; *ТАТ* → *Т*; *ТСА* → *Ц*; *ТАФ* → *ТФ*; *ТС* → *ТЦ*; *ТЦ* → *Ц*; *ТЧ* → *Ч*; *ФАК* → *ФК*; *ФСТФ* → *СТФ*; *ШЧ* → *Ш*;

Example 8. Polyphone will convert the words “ТЕЛЕГРАММА” and “АППАРАТ” into “ТАЛАКРАМА” and “АПАРАТ,” respectively.

For fuzzy word comparison, it is proposed to use the sum of primes assigned to each letter of the converted word: *А* → 2; *П* → 3; *К* → 5; *Л* → 7; *М* → 11; *Н* → 13; *Р* → 17; *С* → 19; *Т* → 23; *У* → 29; *Ф* → 31; *Х* → 37; *Ц* → 41; *Ч* → 43; *Щ* → 47; *Э* → 53; *Я* → 59.

Example 9. The words from the previous example will have the following numerical codes: the word “ТАЛАКРАМА,” the code “71”; the word “АПАРАТ,” the code “49.”

In the paper cited above, Polyphone was compared with other algorithms. In particular, the percentage of correctly identified words in Polyphone was estimated as 95.12 %. The best results for the Russian words transliterated according to GOST (State Standard) R 52535.1-2006 were as follows: in NYSIIS, 75.97%; in SoundEx, 90.24%; in Metaphone, 90.29%; in Double Metaphone, 96.15%; Daitch–Mokotoff Soundex, 96.84%.

At the same time, the accuracy of fuzzy word comparison was declared to reach 98.8%.

3. PHONETIC DISTANCE

The phonetic encoding algorithms considered in the previous section are implemented using the method of equivalent word conversions by sound: a partial word (subword) of a given word belonging to a certain set (equivalence class) is replaced by the code of this set or its typical representative.

Also, note that partial words from the same set, similar in sound, are also similar in spelling. With the introduction of an appropriate metric in words, it is possible to formulate the problem of determining the similarity of words in sound by calculating the distance between words in spelling.

This approach is used in another class of phonetic encoding algorithms: the calculation of the phonetic code of a word is replaced by the pairwise comparison of words by calculating the distance between them in a certain metric space. The speech and writing system of the natural language are assume to have strong correlation with one another. Thus, the problem is find an appropriate metric.

3.1. Levenshtein Distance

The Levenshtein distance is a measure of the difference between two words with respect to the minimum number of insertion, deletion and replacement operations needed to convert one word to another [2].

The Levenshtein distance $L(i,j)$ between two words a and b of length i and j, respectively, is defined as follows: if $\min(i,j)=0$, $L(i,j)=\max(i,j)$; if $\min(i,j)>0$, $L(i,j)$ satisfies the recursive equation

$$L(i,j)=\min(L(i,j-1)+1, L(i-1,j)+1, L(i-1,j-1)+m(i,j)),$$

where $m(i,j)=0$ if the ith letter of the word a coincides with the jth letter of the word b, and $m(i,j)=1$ otherwise.

As it was established in [18], more than 80% of spelling errors are substitution errors. Therefore, the Levenshtein distance is currently determined not in three, but in four operations, namely, insertion, deletion, replacement, and transposition.

The computation algorithm of the Levenshtein distance $L(a, b)$ between words a and b has a complexity of $\Theta(|a| \cdot |b|)$[1] given the memory capacity $\Theta(\min(|a|, |b|))$, where $|x|$ is the length of the word x [80].

A problem caused by determining the phonetic closeness of two words using the Levenshtein distance is the need to set the minimum distance ε between phonetically similar words. To minimize the number of errors, the function ε is chosen dependent on the maximum length n of the compared words. As a rule, $\varepsilon(n < 5) = 0$, $\varepsilon(4 < n < 9) = 1$, and $\varepsilon(8 < n) = 3$.

The functions to compute the Levenshtein distance are implemented in many programming languages [11].

3.2. Distance Based on N-grams

An N-gram is a sequence of N elements (letters). To determine the phonetic closeness of two words, the number of common N-grams is calculated. Usually, N is set equal to 3.

Example 7. Consider two words that have similar sound, namely, "Thomson" and "Thompson." Divide these words into trigrams. As a result, the word "Thomson" includes the trigrams "THO," "HOM," "OMS," "MSO," and "SON"; the word "Thompson," the trigrams "THO," "HOM," "OMP," "MPS," "PSO," and "SON." The common trigrams of these words are "THO," "HOM," and "SON." The share of common trigrams is 3/6, which means that the distance between the words is equal to 3 (the calculation is with respect to the word with a larger number of trigrams.)

Example 8. Consider two other words with similar sound, "Dane" and "Dean." The word "Dane" has the trigrams "DAN" and "ANE"; the word "Dean," the trigrams "DEA" and "EAN." Although these words have similar sound, they have no common trigrams.

As the examples illustrate well, the distance determined using trigrams and the Levenshtein distance are different in the sense that the discriminate letter has a significant effect on the former and a weak effect on the latter. Also, calculating the distance between words based on N-grams obviously gives a better result for longer words than for short ones.

Usually N-grams are used for fuzzy word comparison, which does not affect phonetic aspects. Possible applications of N-grams include language identification (it was established that for sufficiently long texts, each language has its own distribution of N-grams), text compression, guessing subsequent letters, etc. Also, N-grams are widely used for data indexing in search engines [33].

3.3. Jaro Distance

An informal definition of the Jaro distance between two words is the minimum number of one-letter changed required for converting one word to another [30, 31]. The shorter the Jaro distance between the words under comparison is, the greater similarity they have.

The Jaro distance $D(a, b)$ between two words a and b for $m > 0$ has the formula

$$D(a, b) = w_1 \frac{m}{|a|} + w_2 \frac{m}{|b|} + w_3 \frac{m - t}{m},$$

where w_1, w_2, and w_3 are some weight coefficients, $w_1 + w_2 + w_3 = 1$; m gives the number of matching letters (the number of letters spaced no more than half the length of the shortest word); t

[1] The symbol Θ denotes the asymptotic lower and upper bounds, $f(n) \in \Theta(g(n)) \leftrightarrow \exists(C, D, N > 0)\forall(n > N)|C{\cdot}g(n)| \le f(n) \le |D{\cdot}g(n)|$.

denotes half the number of transpositions (half the number of matching letters that differ in serial numbers). For $m = 0$, $D(a, b) = 0$ by definition.

Example 9. Find the Jaro distance between the words "Dane" and "Dean." The number of matching letters m is 4, and half the number of transpositions t is 3/2. With the weight coefficients set equal to 1/3, rounding t to the nearest integer yields $d = 11/12$. In turn, rounding this value to the nearest integer finally gives $d = 1$. Actually, the word "Dean" is converted into the word "Dane" by moving the letter E to the end of the word.

In the paper [81], the formula for calculating the Jaro distance was improved under the assumption that the beginning of a word is more significant compared to the remaining part:

$$d'(a, b) = d(a, b) + s\,[1 - d(a, b)]/10,$$

where $d'(a, b)$ is the improved Jaro distance; s denotes the number of originally common letters of a word, not exceeding four.

Note that the computational complexity of the Jaro distance is the highest of all the previously considered algorithms [47]. As a matter of fact, the lowest computational complexity was observed for the distance calculation algorithm based on N-grams.

4. APPLICATIONS OF PHONETIC ENCODING

Phonetic encoding algorithms are widely used in modern information technology, both in the original and modified forms, for example:

—to normalize data in social networks [9, 52, 68];

—to eliminate duplication of data [6, 49, 54, 57];

—to perform phonetic search [48, 79];

—to translate from one language to another [17];

—to perform search in medical databases [50, 74];

—to introduce string metrics in distributed databases [23];

—to improve Internet search services [3, 69];

—to extract semantic data from ontologies [29];

—to select words of morphological generators [37];

—to identify words during data recovery [82];

—to perform the primary processing of voice queries [72];

—to recognize emergency situations [77];

—to perform speech analysis and synthesis [25, 61];

—to perform data analysis in intelligent systems [35];

—to process and store personal data [24, 39];

—to check spelling in various languages [5, 43, 71];

—to create hashtags in the Big Data technology [13];

—to recognize ethnicity [46];

—to perform statistical entity identification [19];

—to search for spoken documents [66, 67];

—to generate realistic personal data [15];

—to generate hashkeys in data analysis [16];

—to predict customer behavior [61];

—to integrate data based on toponyms [73];

—to index music collections [26, 51];

—to process historical documents [21];

—to correct spelling errors [32, 64];

—to accelerate typing from the keyboard [55];

—to perform the sentiment analysis of users [76];

—to perform cross-language data mining [12];

—to determine the similarity of short messages [63];

—to perform interlanguage transliteration [4, 8, 28];

—to recognize words in text processing [60].

5. CONCLUSIONS

Although the basic phonetic encoding algorithms were proposed in the twentieth century, the research and development of other algorithms never stopped. However, no fundamentally new results were obtained in this field, and all studies actually improved the basic algorithms [34, 65].

Despite all the improvements, the common drawback of all phonetic encoding algorithms is still the presence of false positive and false negative errors in the results. A possible explanation is that these algorithms deal with not the sequence of elementary sounds forming words, but with their textual representation. In addition, textual representations are distorted by the writing rules of one or another natural language.

Obviously, in many cases the existing differences between speech and writing languages make phonetic encoding algorithms incorrect. The phonetic conversion of the words under comparison into a sequence of elementary sounds (allophones, phonemes, diphones, and triphones) seems promising before using the basic phonetic encoding algorithms.

Such an approach can be theoretically based on the recent results established in the field of speech synthesis and recognition. In particular, some algorithms for text voicing and extracting the sequences of elementary sounds from a talk spurt were developed and tested; for details, see [33].

REFERENCES

1. Kankovski, P., “What Is Your Surname?” Or Russian MetaPhone, *Programmist*, 2002, no. 8. pp. 36–39.
2. Levenshtein, V.I., Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, *Dokl. Akad. Nauk SSSR*, 1965, vol. 163, no. 4, pp. 845–848.
3. Abdulhayoglu, M.A. and Thijs, B., Use of ResearchGate and Google CSE for Author Name Disambiguation, *Scientometrics*, Budapest: Akad. Kiado, 2017, vol. 111, pp. 1965–1985.
4. Adeel, Z.M., Iqbal, R.N., and Masood, S.A., English to Urdu Transliteration: An Application of Soundex Algorithm, *IEEE Int. Conf. on Information and Emerging Technologies*, Karachi, Pakistan, June 14–16, 2010, pp. 1–5.
5. Almeida, G., Avanco, L., Duran, M.S., et al., Evaluating Phonetic Spellers for User-Generated Content in Brazilian Portuguese, *Proceedings of the International Conference on Computational Processing of the Portuguese Language*, Tomar, Portugal, June 13–15, 2016, Springer International Publishing, 2016, pp. 361–373.
6. Angeles, M.P. and Perez-Franko, L.F., Analysis of String Encoding Functions During De-Duplication Process, *Proceedings of the International Conference on Informatics, Electronics & Vision*, Fukuaka, Japan, June 15–18, 2015, pp. 1–6.
7. Angeles, M.P., Espino-Gamez, A., and Gil-Moncada, J., Comparison of a Modified Spanish Phonetic, Soundex, and Phonex Coding Functions During Data Matching Process, *International Conference on Informatics, Electronics & Vision*, Fukuaka, Japan, June 15–18, 2015, pp. 1–5.

8. Anonthanasap, O., Ketna, M., and Leelanupab, T., Automated English Mnemonic Keyword Suggestion for Learning Japanese Vocabulary, *Proceedings of the IEEE International Conference on Information Technology and Electrical Engineering (ICITEE)*, Chiang Mai, Thailand, October 29–30, 2015, pp. 638–643.
9. Bilal, A., Lexical Normalization of Twitter Data, *Proceedings of the Science and Information Conference*, London, July 28–30, 2015, pp. 326–328.
10. Binstock, A. and Rex, J., *Practical Algorithms for Programmers*, Boston: Addison-Wesley, 1995.
11. *Calculate Levenshtein Distance between Two Strings.* Available at: http://php.net/manual/en/function.levenshtein.php (Accessed September 8, 2017).
12. Chheda, P., Faruqui, M., and Mitra, P., Handling OOV Words in Indian-Language–English CLIR, *Proceedings of the European Conference on Information*, Barcelona, Spain, April 1–5, 2012, Berlin: Springer-Verlag, 2012, pp. 476–479.
13. Cherichi, S. and Faiz, R., Upgrading Event and Pattern Detection to Big Data, *Proceedings of the International Conference on Computational Collective Intelligence*, Halkidiki, Greece, September 28–30, 2016, Springer International Publishing, 2016, pp. 377–386.
14. Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A., Investigation and Modeling of the Structure of Texting Language, *Int. J. Document Analys. Recognition*, 2007, vol. 10, pp. 157–174.
15. Christen, P. and Pudjijono, A., Accurate Synthetic Generation of Realistic Personal Information, *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Bangkok, Thailand, April 27–30, 2009, Berlin: Springer-Verlag, 2009, pp. 507–514.
16. Christen, P., Geocode Matching and Privacy Preservation, *Proceedings of the International Conference on Privacy, Security and Trust in KDD*, Las Vegas, USA, August 24, 2008, Berlin: Springer-Verlag, 2009, pp. 7–24.
17. Chung, J.M., Lu, C.Y., Lee, H.M., and Ho, J.M., Automatic English-Chinese Name Translation by Using Web-Mining and Phonetic Similarity, *Proceedings of the IEEE International Conference on Information Reuse & Integration*, Las Vegas, USA, August 3–5, 2011, pp. 283–287.
18. Damerau, F.J., A Technique for Computer Detection and Correction of Spelling Errors, *ACM*, 1964, no. 7(3), pp. 171–176.
19. Denk, M., Framework for Statistical Entity Identification in R, in *Data Analysis, Machine Learning and Applications*, Berlin: Springer-Verlag, 2008, pp. 335–342.
20. Donghui, L. and Dewei, P., Spelling Correction for Chinese Language Based on Pinyin–Soundex Algorithm, *Proceedings of the International Conference on Internet Technology and Applications*, Wuhan, China, August 16–18, 2011, pp. 1–3.
21. Ernst-Gerlach, A. and Fuhr, N., Advanced Training Set Construction for Retrieval in Historic Documents, *Proceedings of the Asia Information Retrieval Symposium*, Taipei, Taiwan, December 1–3, 2010, Berlin: Springer-Verlag, 2010, pp. 131–140.
22. Gaona, M.A., Gelbukh, A., and Bandyopadhyay, S., Recognizing Textual Entailment Using a Machine Learning Approach, *Proceedings of the Mexican International Conference on Artificial Intelligence*, Pachuca, Mexico, November 8–13, 2010, Berlin: Springer-Verlag, 2010, pp. 177–185.
23. Giraud-Carrier, C., Goodliffe, J., Jones, B.M., and Cueva, S., Effective Record Linkage for Mining Campaign Contribution Data, in *Knowledge and Information Systems*, London: Springer-Verlag, 2015, vol. 45, pp. 389–416.
24. Grzebala, P. and Cheatham, M., Private Record Linkage: Comparison of Selected Techniques for Name Matching, *Proceedings of the International Semantic Web Conference*, Heraklion, Crete, Greece, May 29–June 2, 2016, Springer International Publishing, 2016, pp. 593–606.
25. Jian, H.-L., Speech Driven by Artificial Larynx: Potential Advancement Using Synthetic Pitch Contours, *Proceedings of the International Conference on Universal Access in Human-Computer Interaction*, Los Angeles, USA, August 2–7, 2015, Springer International Publishing, 2015, pp. 312–321.
26. Han, Y., Min, L., Zou, Y., et al., LRC Sousou: A Lyrics Retrieval System, *Proceedings of the International Conference of Young Computer Scientists, Engineers and Educators*, Harbin, China, January 10–12, 2015, Berlin: Springer-Verlag, 2015, pp. 464–467.

27. Herzog, T.N., Scheuren, F.J., and Winkler, W.E., *Data Quality and Record Linkage Techniques*, New York: Springer, 2007, pp. 115–121.

28. Hong, S.G., Jang, S., Chung, Y.H., et al., News Media Analysis Using Focused Crawl and Natural Language Processing: Case of Lithuanian News Websites, *Proceedings of the International Conference on Information and Software Technologies*, Kaunas, Lithuania, September 13–14, 2012, Berlin: Springer-Verlag, 2012, pp. 48–61.

29. Hu, B. and Hu, B., On Capturing Semantics in Ontology Mapping, *World Wide Web*, 2008, vol. 11, pp. 361–385.

30. Jaro, M.A., UNIMATCH—A Computer System for Generalized Record Linkage under Conditions of Uncertainty, *Proceedings of the Spring Joint Computer Conference*, Anaheim, USA December 5–17, 1972, pp. 523–530.

31. Jaro, M.A., Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida, *J. Am. Statist. Ass.*, 1989, no. 84(406). pp. 414–420.

32. Jordao, C.C. and Rosa, J.L., Metaphone-pt_BR: The Phonetic Importance on Search and Correction of Textual Information, *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*, New Delhi, India, March 11–17, 2012, Berlin: Springer-Verlag, 2012, pp. 297–305.

33. Jurafsky, D. and Martin, J.H., *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Pearson: Prentice Hall, 2009.

34. Karakasidis, A. and Verykios, V.S., Privacy Preserving Record Linkage Using Phonetic Codes, *Proceedings of the Balkan Conference in Informatics*, Thessaloniki, Greece, September 17–19, 2009, pp. 101–106.

35. Karakasidis, A., Koloniari, G., and Verykios, V.S., Privacy Preserving Blocking and Meta-Blocking, *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Porto, Portugal, September 7–11, 2015, Springer International Publishing, 2015, pp. 232–236.

36. Kaveh-Yazdy, F. amd Zareh-Bidoki, A.M., Aleph or Aleph-Maddah, That Is the Question! Spelling Correction for Search Engine Autocomplete Service, *Proceedings of the International Conference on Computer and Knowledge*, Mashhad, Iran, October 29–30, 2014, pp. 273–278.

37. Knuth, D.E., *The Art of Computer Programming*, Boston: Addison-Wesley, 1998, vol. 3.

38. Koco, J. and Piasecki, M., Named Entity Matching Method Based on the Context-Free Morphological Generator, *Proceedings of the International Conference on Natural Language Processing*, Warsaw, Poland, September 17–19, 2014, Springer International Publishing, 2014, pp. 34–44.

39. Kroll, M. and Steinmetzer, S., Who Is 1011011111...1110110010? Automated Cryptanalysis of Bloom Filter Encryptions of Databases with Several Personal Identifiers, *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies*, Lisbon, Portugal, January 12–15, 2015, Springer International Publishing, 2015, pp. 341–356.

40. Lawrence, P., Hanging on the Metaphone, *Comput. Language*, 1990, vol. 7, no. 12, pp. 39–44.

41. Lawrence, P., The Double Metaphone Search Algorithm, *C/C++ Users J.*, 2000, no. 18(6), pp. 38–43.

42. Lawrence, P., *Metaphone 3: Version 2.1.3*. https://searchcode.com/codesearch/view/2366000/ (Accessed February 18, 2012).

43. Li, J., Ouazzane, K., Jing, Y., et al., Evolutionary Ranking on Multiple Word Correction Algorithms Using Neural Network Approach, *Proceedings of the International Conference on Engineering Applications of Neural Networks*, London, August 27–29, 2009, Berlin: Springer-Verlag, 2009, pp. 409–418.

44. Li, L., Quazzane, K., Kazemian, H., Jing, Y., and Boyd, R., A Neural Network Based Solution for Automatic Typing Errors Correction, *Neural Comput. Appl.*, 2011, vol. 20, pp. 889–896.

45. Lisbach, B. and Meyer, M., *Linguistic Identity Matching*, Wiesbaden: Springer Fachmedien Wiesbaden, 2013, pp. 118–120.

46. Louppe, G., Al-Natsheh, H.T., Susik, M., and Maguire, E.J., Ethnicity Sensitive Author Disambiguation Using Semi-Supervised Learning, *Proceedings of the International Conference on Knowledge Engineering and the Semantic Web*, Prague, Czech Republic, September 21–23, 2016, Springer International Publishing, 2016, pp. 272–287.

47. Maarif, H.A., Akmeliawati, R., Htike, Z.Z., and Gunawan, T.S., Complexity Algorithm Analysis for Edit Distance, *Proceedings of the International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia, September 23–24, 2014, pp. 135–137.

48. Mandal, A.K., Hossan, D., and Nadim, M., Developing an Efficient Search Suggestion Generator, Ignoring Spelling Error for High Speed Data Retrieval Using Double Metaphone Algorithm, *Proceedings of the International Conference on Computer and Information Technology*, Dhaka, Bangladesh, December 23–25, 2010, pp. 317–320.

49. Martins, B., A Supervised Machine Learning Approach for Duplicate Detection over Gazetteer Records, *Proceedings of the International Conference on GeoSpatial Semantics*, Brest, France, May 12–13, 2011, Berlin: Springer-Verlag, 2011, pp. 34–51.

50. Mason-Blakley, F., Lu, L., Price, M., and Roudsari, A., An RCT Simulation Study on Performance and Accuracy of Inexact Matching Algorithms for Patient Identity in Ambulatory Care Settings, *Proceedings of the International Conference on Healthcare Informatics*, Dallas, USA, October 21–23, 2015, pp. 8–17.

51. Meintanis, K. and Shipman, F.M., Visual Expression for Organizing and Accessing Music Collections in MusicWiz, *Proceedings of the International Conference on Theory and Practice of Digital Libraries*, Glasgow, UK, September 6–10, 2010, Berlin Heidelberg: Springer-Verlag, 2010, pp. 80–91.

52. Mosquera, A. and Moreda, P., The Study of Informality as a Framework for Evaluating the Normalisation of Web 2.0 Texts, *Proceedings of the International Conference on Application of Natural Language to Information Systems*, Groningen, The Netherlands, June 26–28, 2012, Berlin Heidelberg: Springer-Verlag, 2012, pp. 241–246.

53. Mutalib, A. and Noah, S.A., Phonetic Coding Methods for Malay Names Retrieval, *Proceedings of the IEEE International Conference on Semantic Technology and Information*, Putrajaya, Malaysia, June 28–29, 2011, pp. 125–129.

54. Muthmann, K. and Loser, A., Detecting Near-Duplicate Relations in User Generated Forum Content, *Proceedings of the OTM Confederated International Conference "On the Move to Meaningful Internet Systems,"* Hersonissos, Crete, Greece, October 25–29, 2010, Berlin: Springer-Verlag, 2010, pp. 698–707.

55. Ouazzane, K., Li, J., and Kazemian, H.B., An Intelligent Keyboard Framework for Improving Disabled People Computer Accessibility, *Proceedings of the International Conference on Engineering Applications of Neural Networks*, International Federation for Information Processing, Corfu, Greece, September 15–18, 2011, pp. 382–391.

56. Ousidhoum, N.D. and Bensaou, N., Towards the Refinement of the Arabic Soundex, *Proceedings of the International Conference on Application of Natural Language to Information Systems*, Salford, UK, June 19–21, 2013, Berlin: Springer-Verlag, 2013, pp. 309–314.

57. Owonibi, M. and Koenig-Ries, B., A Quality Management Workflow Proposal for a Biodiversity Data Repository, *Proceedings of the International Conference on Conceptual Modeling*, Atlanta, USA, October 27–29, 2014, in *Advances in Conceptual Modeling*, Lecture Notes in Computer Science, vol. 8823, Switzerland: Springer International Publishing, 2014, pp. 157–167.

58. Pande, B.P. and Dhami, H.S., Application of Natural Language Processing Tools in Stemming, *Int. J. Computer Appl.*, 2011, no. 27(6), pp. 14–19.

59. Paramonov, V.V., Shigarov, A.O., Ruzhnikov, G.M., et al., Polyphon: An Algorithm for Phonetic String Matching in Russian Language, *Proceedings of the International Conference on Information and Software Technologies*, Druskininkai, Lithuania, October 13–15, 2016, Springer International Publishing, 2016, pp. 568–579.

60. Parmar, V.P., Pandya, A.K., and Kumbharana, C.K., Determining the Character Replacement Rules and Implementing Them for Phonetic Identification of Given Words to Identify Similar Pronunciation Words, *Proceedings of the IEEE International Conference on Futuristic Trends in Computational Analysis and Knowledge Management (ABLAZE)*, New Delhi, India, February 25–27, 2015, pp. 272–277.

61. Parvathy, A.G, Vasudevan, G.B., Kumar, A., and Balakrishnan, R., Leveraging Call Center Logs for Customer Behavior Prediction, *Proceedings of the International Symposium on Intelligent Data Analysis*, New Delhi, India, February 25–27, 2009, Berlin: Springer-Verlag, 2009, pp. 143–154.

62. Pfeiffer, A., Kuraeva, A., Foulonneau, M., et al. Automatically Generated Metrics for Multilingual Inline Choice Questions on Reading Comprehension, *Proceedings of the International Computer Assisted*

Assessment Conference, Zeist, The Netherlands, June 22–23, 2015, Springer International Publishing, 2015, pp. 80–95.

63. Pinto, D., Vilarino, D., Aleman, Y., et al. The Soundex Phonetic Algorithm Revisited for SMS Text Representation, *Proceedings of the International Conference on Text, Speech and Dialogue*, Brno, Czech Republic, September 3–7, 2012, Berlin: Springer-Verlag, 2012, pp. 47–55.
64. Purao, S., Storey, V.C., Sugumaran, V., et al., Repurposing Social Tagging Data for Extraction of Domain-Level Concepts, *Proceedings of the International Conference on Application of Natural Language to Information Systems*, Alicante, Spain, June 28–30, 2011. Berlin: Springer-Verlag, 2011, pp. 185–192.
65. Rakibul, H. and Reaz, A., FM-Chord: Fault-tolerant Chord Supporting Misspelled Queries, *Proceedings of the International Conference on Computers and Information Technology*, Dhaka, Bangladesh, December 21–23, 2009, pp. 651–656.
66. Reyes-Barragan, M., Villasenor-Pineda, L., and Montes-Y-Gomez, M., A Soundex-Based Approach for Spoken Document Retrieval, *Proceedings of the Mexican International Conference on Artificial Intelligence*, Atizapan de Zaragoza, Mexico, October 27–31, 2008, Berlin: Springer-Verlag, 2008, pp. 204–211.
67. Reyes-Barragan, A., Montes-Y-Gomez, M., and Villasenor-Pineda, L., Combining Word and Phonetic-Code Representations for Spoken Document Retrieval, *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*, Tokyo, Japan, February 20–26, 2011, Berlin: Springer-Verlag, 2011, pp. 458–466.
68. Roedler, R., Kergl, D., and Rodosek, G.D., Profile Matching Across Online Social Networks Based on Geo-Tags, in *Advances in Nature and Biologically Inspired Computing*, Springer International Publishing, 2016, pp. 417–428.
69. Russell, R.C. and Margaret, K.O., *US Patent 1262167, 1435663*, 1918, 1922.
70. Sanchez-Vilas, F., Lama, M., Vidal, J.C., et al., Combining Uncorrelated Similarity Measures for Service Discovery, *Proceedings of the International Workshop on Resource Discovery*, Paris, France, November 5, 2010, Berlin: Springer-Verlag, 2012, pp. 160–180.
71. Schierle, M., Schulz, S., and Ackermann, M., From Spelling Correction to Text Cleaning—Using Context Information, in *Data Analysis, Machine Learning and Applications*, Berlin: Springer-Verlag, 2008, pp. 397–404
72. Schneider, J.M., Fernandez, J., and Martinez, P., A Proof-of-Concept for Orthographic Named Entity Correction in Spanish Voice Queries, *Proceedings of the International Workshop on Adaptive Multimedia Retrieval*, Copenhagen, Denmark, October 24–25, 2012, Springer International Publishing, 2014, pp. 181–190.
73. Smart, P.D., Jones, C.B., and Twaroch, F.A., Multi-source Toponymal Data Integration and Mediation for a Meta-Gazetteer Service, *Proceedings of the International Conference on Geographic Information Science*, Zurich, Switzerland, April 30, 2010, Berlin: Springer-Verlag, 2010, pp. 234–248.
74. Soman, S., Srivastava, P., and Murthy, B.K., Unique Health Identifier for India: An Algorithm an Feasibility Analysis on Patient Data, *Proceedings of the International Conference on E-health Networking, Application & Services*, Boston, USA, October 13–17, 2015, pp. 250–255.
75. *Soundexing and Genealogy by Gary Mokotoff*. www.avotaynu.com/soundex.htm (Accessed September 8, 2017).
76. Souza, M. and Vieira, R., Sentiment Analysis on Twitter Data for Portuguese Language, *Proceedings of the 10th International Conference on Computational Processing of the Portuguese Language (PROPOR'12)*, Berlin: Springer-Verlag, 2012, pp. 241–247.
77. Taft, R.L., *Name Search Techniques*, New York State Identification and Intelligence System: Special Report No. 1, New York: Albany, 1970.
78. Souza, J., Botega, L.C., Segundo, S., et al., Conceptual Framework to Enrich Situation Awareness of Emergency Dispatchers, *Proceedings of the International Conference on Human Interface and the Management of Information*, Los Angeles, USA, August 2–7, 2015, Springer International Publishing, 2015, pp. 33–44.
79. Tissot, H., Peschl, G., and Fabro, M.D., Fast Phonetic Similarity Search over Large Repositories, *Proceedings of the International Conference on Database and Expert Systems Applications*, Munich, Germany, September 1–4, 2014, Springer International Publishing, 2014, pp. 74–81.

80. Wagner, R.A. and Fischer, M.J., The String-to-String Correction Problem, *ACM*, 1974, no. 21(1), pp. 168–173.

81. Winkler, W.E., String Comparator Metrics and Enhanced Decision Rules in the Fellegi–Sunter Model of Record, *Proc. of the Section on Survey Research Methods*, American Statistical Association, 1990, pp. 354–359.

82. Zampieri, M. and Amorim, R.C., Between Sound and Spelling: Combining Phonetics and Clustering Algorithms to Improve Target Word Recovery, *Proceedings of the International Conference on Natural Language Processing*, Warsaw, Poland, September 17–19, 2014, Springer International Publishing, 2014, pp. 438–449.

This paper was recommended for publication by ???, a member of the Editorial Board